Anno IX - N. 73 - 25 Aprile 1990 - Sped. Abb. Post. III/70 - CR - Distr.: Parrini

Sempre più veloce col 128

AMIGA

L'acchiappa-grafica

Trasferimento Ascii da 64 ad Amiga

C 64

- *Assicurazioni*
- *variabili locali*

Leggo VR perché mi dà la rotta

Il lettore di VR è giovane, dinamico, creativo. Di cultura e reddito superiore alla media, possiede spesso più di un videoregistratore, oltre all'impianto hi-fi e al computer: nel tempo libero, non rinuncia ai viaggi in Italia e all'estero, e a cinema, teatro e spettacoli sportivi in genere. Usa il videoregistratore non solo per i programmi tv o preincisi, ma anche per riprendere i momenti felici in famiglia, per creare una videoteca personale.
E tu, che tipo di lettore sei?

VR
VIDEOREGISTRARE

IMMAGINE MARSILII

Sommario

RUBRICHE

Sempre più veloce col 128

Direttore: Alessandro de Simone - **Coordinatore:** Marco Miotti
Redazione/collaboratori: Paolo Agostini, Davide Ardizzone, Claudio Baiocchi, Angelo Bianchi, Luigi Callegari, Sergio Camici, Umberto Colapicchioni, Laura & Miria Colombo, Valerio Ferri, Simona Locati, Michele Maggi, Giancarlo Mariani, Clizio Merli, Marco Mietta, Marco Miotti, Oscar Moccia, Roberto Morassi, Guido Pagani, Antonio Pastorelli, Domenico Pavone, Armando Sforzi, Sonja e Patrizia Scharrer, Dario Pistella, Fabio Sorgato, Valentino Spataro, Danilo Toma, Franco Rodella, Stefano Simonelli
Grafica: Arturo Ciaglia
Direzione, pubblicità: via Mosè, 22 - 20090 Opera (MI) - Tel. 02/55500310 - Fax 02/57603039 - **Redazione:** Tel. 02/55500310
Pubblicità: Milano: Leandro Nencioni (direttore vendite), - Via Mosè, 22 - 20090 Opera (MI) - Tel. 02/55500310
• Emilia Romagna: Spazio E - P.zza Roosevelt, 4 - 40123 Bologna - Tel. 051/236979
• Toscana, Marche, Umbria: Mercurio srl - via Rodari, 9 - San Giovanni Valdarno (Ar) - Tel. 055/947444
• Lazio, Campania: Spazio Nuovo - via P. Foscari, 70 - 00139 Roma - Tel. 06/8109679
Segreteria: Marina Vantini - **Abbonamenti:** Liliana Spina
Arretrati e software: Opera, Via Mosè, 22 - tel. 02/55500310 - Sig. ra Lucia Dominoni (Il servizio è operativo nelle ore pomeridiane.)
Tariffe: prezzo per copia L. 6.000. Abbonamento annuo (11 fascicoli) L. 50.000. Estero: il doppio.
Abbonamento cumulativo alle riviste Computer e Commodore Computer Club L. 90.000.
I versamenti vanno indirizzati a: Systems Editoriale Srl mediante assegno bancario o utilizzando il c/c postale n. 37952207
Composizione: Systems Editoriale Srl - **Fotolito:** Systems Editoriale Srl
Stampa: Systems Editoriale/La Litografica Srl - Busto Arsizio (Va)
Registrazioni: Tribunale di Milano n. 370 del 2/10/82 - Direttore Responsabile: Michele Di Pisa
Sped. in abb. post. gr. III - Pubblicità inferiore al 70% - **Distrib.:** Parrini - Milano
Periodici Systems: Banca Oggi - Commodore Club (disco) - Commodore Computer Club - Commodore Computer Club (disco produzione tedesca) - Computer - Computer disco - Electronic Mass Media Age - Energy Manager - Hospital Management - Jonathan - Nursing '90 - PC Programm (disco) - Personal Computer - Security - Software Club (cassetta ed. italiana) - TuttoGatto - Videoteca - VR Videoregistrare

LA VOSTRA POSTA

QUALE DRIVE

* Ho intenzione di "espandere" il mio C/64, già dotato di registratore e stampante, con un drive. Sono però indeciso tra un modello originale o compatibile e tra il formato 5.25 e 3.5. Potete aiutarmi nella scelta?

(Ermanno Salvalaio - Maerne)

* Anzitutto è bene precisare che l'unico drive in formato 3.5 pollici disponibile per il C/64 è quello originale Commodore, modello *1581*; è altrettanto utile sottolineare, onde evitare spiacevoli delusioni, che tale periferica è consigliata (quasi) esclusivamente a chi già possiede un drive di formato 5.25 pollici e desidera entrare in possesso di una periferica più veloce, versatile e capiente.

Il 1581, infatti, è decisamente più veloce del 1541, offre una capienza più che quadrupla e la possibilità di gestire subdirectory. Le caratteristiche, di tutto rilievo, possono essere apprezzate soprattutto da chi utilizza programmi professionali (*word processor, spread-sheet* e *data base*) con una certa intensità, a patto, ovviamente, che questi siano sufficientemente... sprotetti da consentire il loro "trasporto" sul dischetto da 3.5.

Non dimentichiamo, infatti, che non solo tutto il software professionale (originale) per C/64 è disponibile su dischetti da 5.25, ma che i vari *package* spesso (= quasi sempre) non consentono di "scaricare" i dati su periferiche diverse dal n. 8.

Ciò significa, in pratica, l'impossibilità di usare il 1581 con software originale dal momento che sul 1541 deve esser presente il disco contenente il programma (in formato 5.25).

Se il software può, invece, esser trasferito senza problemi su dischetti da 3.5 non sussistono le difficoltà citate.

E passiamo ora al secondo, annoso problema: originale o compatibile?

Conosco dozzine di soddisfatti utenti di 1541 "*compatibili*" che, risparmiando un buon 30%, sono entrati in possesso di una periferica che si comporta in modo egregio.

Molti utenti, però, lamentano la non totale compatibilità con il 1541 originale, soprattutto con alcuni giochi protetti in modo davvero insolito.

A creare confusione provvede la stessa Commodore che, con varie azioni legali, ha tentato (e tenta tuttora) di impedire la vendita dei drive compatibili sostenendo che le Rom risultano rigorosamente identiche (cioè copiate) a quelle del 1541; con tale comportamento, del tutto legittimo, la Commodore non si accorge, però, di aiutare indirettamente la "concorrenza" che approfitta delle vicende giudiziarie per sostenere la tesi della totale compatibilità dei drive commercializzati. Se questi, infatti, non fossero totalmente compatibili, sostengono i commercianti dei *clone*, la Commodore non si prenderebbe la briga di sporger denuncia.

Non per ultima, poi, dovrebbe esser considerata l'eventuale assistenza in caso di guasti. E' difficile, infatti, stabilire chi si comporta meglio, almeno a giudicare da alcune terrificanti lettere giunte in Redazione, a tal proposito, da esasperati lettori.

Quale drive, dunque, comprare?

Ai lettori l'ardua sentenza...

CORREZIONE

* A mio parere, il programma per C/64 a corredo dell'articolo "***Oltre la musica del SID***" (CCC n. 61) presenta un errore. Per fare in modo che la freccia venga gestita dal joystick, occorre modificare la riga 330 come segue:

```
330 co = 32: rg = z: gosub 440: rg = y:
print " ": gosub 440: print chr$ (95): z = y
```

(Vittorio Benassi - Correggio)

* Il programma ***Koala Video*** (CCC n. 27, pag. 81) che consente di esaminare schermate in formato Koala, contiene un piccolo errore che causa l'alterazione dei colori. E' necessaria una semplice modifica per eliminare il difetto:

12036 Poke 53280, 0: Poke 53281, Peek (34576)

(Claudio Fontacci - Roma)

* Ringraziamo per le cortesi segnalazione.

MACROASSEMBLER E COBOL

* Non riesco a trovare, nemmeno presso i ***Commodore Point***, il Macro Assembler Commodore nè il Cobol, per C/64. Come posso fare per entrarne in possesso?

(Daniel Ferrato - Como)
(Francesco Maggio - Palermo)

* Siamo alle solite: alcune Ditte si lamentano della pirateria, però non fanno nulla per limitarla. C'è da dire, a parziale discolpa della Commodore, che il linguaggio *Cobol* per C/64 non è marchiato Commodore (a differenza del *Macro Assembler*).

Per quanto riguarda il modo di procurarsi il software che ti occorre, è bene sottolineare che è rigorosamente vietato entrare in contatto sia con banche dati (che spesso offrono s/w "particolare" via telefono), sia con lettori che pubblicano annunci economici sia con negozi che,

dietro segnalazione, riescono a procurare copie dei programmi richiesti...

Commodore Computer Club risponde a tutte le domande dei suoi lettori. E' ovvio che avranno precedenza le domande di interesse generale che possano aiutare a risolvere i problemi della maggior parte degli utenti Commodore. Non si risponde alle domande la cui risposta può facilmente essere rintracciata nei libretti di istruzione degli apparecchi. Si consiglia, pertanto, di leggere con la massima attenzione i manuali a corredo di computer, drive, stampanti, monitor (e così via) prima di inviare le vostre lettere.
Ricordiamo che l'indirizzo presso cui inviare la corrispondenza è:

Commodore Computer Club
Rubrica "La posta"
Via Mosè, 22
20090 OPERA (Mi)

RADICI CUBICHE

* Come posso estrarre, con il mio C/64, la radice cubica di un numero?
(Luca Garulli - Grosseto)

* La radice di un numero non è altro che una particolare potenza.

Per estrarre la radice cubica di un numero, insomma, è sufficiente elevare il numero in questione alla potenza dell'inverso di tre. In altre parole...

radice cubica di 27 = 27 ^ (1/3)

radice ottava di 98 = 98 ^ (1/8)

...e così via.

Tale "trucchetto" matematico è valido per qualsiasi altro computer e/o linguaggio di programmazione. Anche con Amiga, infatti, vale lo stesso discorso.

La presenza dell'istruzione *Sqr*, valida per determinare la radice quadrata di un numero, deve esser considerata come un *extra* dal momento che la regola appena citata vale anche per la radice quadrata.

Radice quadrata di 16, infatti, è eguale a...

16 ^ (1/2)

Altre relazioni matematiche, non disponibili "direttamente", possono egualmente essere adattate al linguaggio Basic con la massima semplicità.

DERATTIZZAZIONE

* Ho notato, ultimamente, una graduale scomparsa dei simpaticissimi topolini che pullulavano fra le pagine di CCC. A che cosa è dovuta la decisione di limitarne la pubblicazione?
(Annalisa ed altri lettori)

* Il mouse (= topolino) rappresenta ormai lo strumento di *Input* più popolare, comodo e versatile a disposizione dell'*homo informaticus*. Qualsiasi computer moderno non può farne a meno: lo ha dimostrato il *Mcintosh*, *l'Amiga* e *l'Atari* (che lo forniscono di serie, come facente parte integrante ed insostituibile del sistema) e perfino gli elaboratori Ms-Dos compatibili che, finalmente, si sono arresi di fronte al successo riscontrato dal minuscolo, quanto prezioso, accessorio.

Il topolino che compariva sulle nostre pagine voleva rappresentare, appunto, il simbolo stesso dell'informatica di massa, ma non sempre ha suscitato reazioni positive tra i lettori.

Come di consueto, "ovviamente", cominciano a giungere in Redazione le lamentele di coloro che, abituati ai simpatici topi, vorrebbero un loro ritorno.

Chi dobbiamo accontentare?

DUE COMPUTER

* Ho due C/64 e vorrei sapere se, collegandoli in qualche modo tra loro, posso ottenere qualche vantaggio, tipo disporre di più memoria o usare particolari programmi che prevedano, appunto, l'utilizzo contemporaneo di due computer.
(Massimo Stacchiotti - Iesi)

* Due computer vengono collegati tra loro per scambiar dati reciprocamente.

Con il primo, ad esempio, trasmetti un programma e con il secondo lo ricevi. Naturalmente si potrebbe obiettare che è molto più semplice (e comodo) registrare il programma su un dischetto che, in seguito, può esser "letto" dal secondo computer.

In effetti è proprio così!

Un collegamento tra due computer, quindi, ha senso se i calcolatori sono distanti parecchi chilometri tra loro (e si utilizza il modem, in questo caso) oppure appartengono a standard diversi tra loro; in quest'ultimo caso è possibile trasferire un file Ascii (come una lettera o un documento in genere) tra un C/64 ed un computer Ms-Dos grazie alla porta di interfaccia Rs-232. Usufruire di una maggiore "potenza" sembrerebbe possibile collegando due C/64 tra loro; ma si presenta subito il problema di scrivere software specifico dal momento che, a quanto mi risulta, non esistono programmi che prevedono simili collegamenti.

Per l'Amiga, invece, sono disponibili numerosissimi programmi (soprattutto videogames) che prevedono esplicitamente la connessione tra due elaboratori.

I più famosi sono i simulatori di volo grazie ai quali ciascun giocatore pilota il suo aereo in modo perfettamente indipendente dall'altro. Quando, però, i due apparecchi si trovano nello stesso spazio aereo, è possibile ingaggiare una battaglia. In questo caso ciascuno dei due giocatori vedrà, ognuno sul proprio video, l'apparecchio del "nemico", con un realismo difficilmente immaginabile se si è abituati a governare due aerei (ognuno collegato ad un joy) presenti contemporaneamente su di un unico schermo.

ESISTENZA IN VITA

* Come si può controllare se, su un dischetto, è presente un certo file?
(Valerio Granato - Napoli)

* Sul drive 1541 è disponibile un canale privilegiato, individuabile dal n. 15, che si incarica, appunto, di esaminare eventuali errori che dovessero capitare nel "colloquio" con il computer.

E' sufficiente aprire il suddetto canale in occasione di qualsiasi gestione di file e, nei casi più delicati, interrogare la periferica sullo scambio di dati in corso.

Il seguente programmino...

```
100 input "nome del file";nf$
```

```
110 open 15, 8, 15
120 open 1, 8, 8, nf$: gosub 160
130 if x=1 then 100
140 print "il file " nf$ " esiste"
150 close 1: close 15: end
160 input# 15, a
170 x = 0: if a = 62 then 190
180 return
190 print "il file " nf$;
200 print " non esiste"
210 x = 1: close 1: close 15: return
```

...chiede (riga 100) il nome del file di cui si vuol verificare la presenza su disco.

Viene quindi aperto il canale 15 (per il futuro controllo) e, in riga 120, un canale in lettura (che, quindi, non modificherà in alcun modo il file eventualmente presente).

Subito dopo si salta alla riga 160 che legge (*input*) il primo dato disponibile sul drive; questo (variabile *A*) dovrebbe esser sempre posto a zero; in caso contrario vuol dire che c'è qualche inconveniente.

Nel caso specifico il valore 62 indica l'assenza del file cercato (gli altri valori / codice sono rintracciabili nell'appendice del manuale di istruzioni del 1541).

Contemporaneamente la variabile / deviatore *X* provvede, al ritorno dalla subroutine, a visualizzare il messaggio opportuno ed a riformulare la domanda (*goto 100*) oppure a terminare il programma.

Inutile dire che le notizie qui riportate sono pubblicate anche sul manuale di istruzioni: non era meglio leggerlo invece di scrivere la lettera?

JOYSTICK POCO CHIARO

* Ho progettato un joy particolare(di cui allego alcune foto) che consente di manovrare con maggior comodità i personaggi e gli oggetti di numerosi videogames.

Ritenete che il progetto sia di un certo interesse per i lettori di Commodore Computer Club?

(Paolo Besser - Vigevano)

* Penso di sì, a patto di avere a disposizione foto dignitose (e non sfocate e riprese in una giornata di nebbia a Linate) oppure (meglio) uno schema elettrico disegnato con righello ed inchiostro di china oltre ad un articolo che descriva con precisione la funzione svolta dai vari componenti impiegati.

L'articolo puoi scriverlo con il popolare word processor *Superscript 128*, che dichiari di possedere.

SUPERLOTTO

* Ho scritto un programma per il gioco del Lotto che, grazie ad una tecnica innovativa (che non ricorre alla solita "gestione" dei ritardi), permette di escludere moltissimi dei 90 numeri estraibili allo scopo di puntare su una ristretta cerchia di valori. Ritenete che il programma possa esser pubblicato?

(Nazareno Signoretto - Cerea)

* Sono ancora fermamente convinto che il gioco del Lotto, per la particolare forma di "premiazione" prevista, rappresenti una vera e propria truffa ai danni della popolazione.

Tuttavia non posso fare a meno di riconoscere che, nonostante tutto, sia uno dei giochi più popolari in Italia.

E' probabile che la *Systems Editoriale* sia disponibile per la pubblicazione di un numero "speciale" su supporto magnetico, ma il compenso dovrebbe esser molto, molto, ma molto inferiore a quello richiesto nella lettera.

Si faccia sentire, ne possiamo riparlare.

SUPERMASTERMIND

* Ho scritto, con il mio C/128, il programma Super - Master - Mind che presenta difficoltà crescenti fino a nove cifre da indovinare.

Può interessare per un'eventuale pubblicazione?

(Alessio G. - Roma)

* Purtroppo i programmi semplici come Master Mind (o Super che siano) hanno fatto il loro tempo e non offrono più il divertimento che, qualche anno fa, era garantito da programmi del genere.

DRIVER GEOS PER 1230

* Penso di essere in grado di aiutare quel lettore che, usando Geos, aveva difficoltà ad usare la stampante Mps-1230.

E' infatti necessario modificare la predisposizione della stampante all'accensione. Bisogna, pertanto, selezionare la modalità "Epson FX-80", e sostituire "open - mode" con 4 P.C. Commands (5 Commodore commands); il driver Epson è presente sul lato B di Geos.

(Luca Del Monego - Alleghe)

ASCENDENTI E DISCENDENTI

* Ho acquistato, di recente, un sistema usato completo (C/64, 1541, Mps-803) di cui sono soddisfatto.

Mi piacerebbe, però, generare caratteri discendenti ed ascendenti che migliorerebbero la resa grafica su carta.

(Mauro Belfiori - Roma)

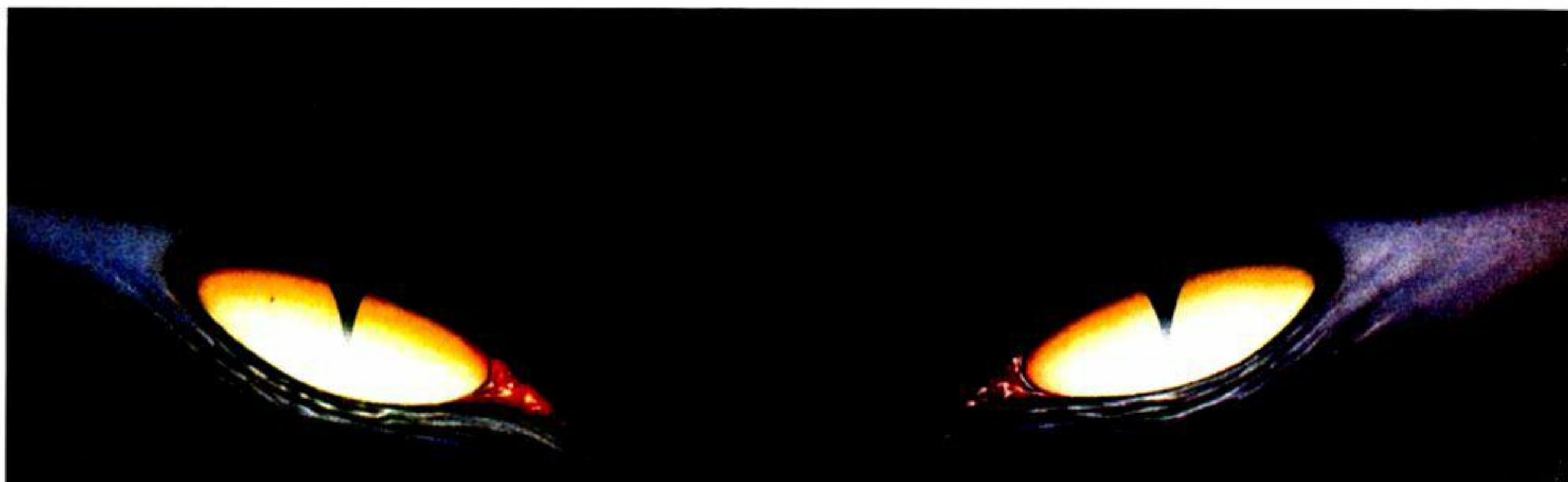
* E' possibile effettuare la modifica richiesta, ma è necessario sostituire la Rom della stampante portando la periferica presso un negozio specializzato che offra tale servizio.

NON INVIATE FRANCOBOLLI

La corrispondenza di Commodore Computer Club viene evasa con la massima sollecitudine possibile privilegiando gli argomenti che possano attirare l'interesse della maggior parte dei lettori.

Non sono dunque previste risposte "personali" a quesiti molto specifici; questi possono tuttavia esser posti per telefono (02/52.49.211) in orari preferibilmente pomeridiani.

Si prega, pertanto, di non insistere per ottenere informazioni secondo modalità diverse da quelle appena esposte.

SCOPRI I MISTERI DEI VIDEOGAMES

Ti hanno tolto il gusto di creare videogiochi?
Quelli che vedi in commercio (e di cui sicuramente possiedi alcuni esemplari) sono talmente strabilianti, ricchi di schermate e, in definitiva, impossibili da realizzare se non a costo di estenuanti ore passate a programmare?

Bene, puoi egualmente dimostrare a tutti la tua abilità nel risolvere i giochi più difficili, in modo da costringere i programmatori professionisti a realizzare videogames sempre più impegnativi, avvincenti e... a prova di ficcanaso.

ORA TOCCA A TE!

Se hai scoperto la **mappa** di un gioco di avventura, se hai individuato le **Poke** che aumentano le "vite" all'infinito, se hai avuto la fortuna (o, diciamolo senza falsi pudori, l'abilità) di "entrare" nel programma per modificare i nomi degli autori, il **punteggio** massimo raggiunto, il sistema per raggiungere subito le **schermate** finali (o altre diavolerie del genere), ebbene ora tocca a te!

Dimostra a tutti la tua bravura inviando la soluzione del gioco che hai "violato" o i vari trucchetti scoperti, sia che tu li abbia individuati sulla versione originale sia su quella pirateggiata.

Se ciò che invierai corrisponderà al vero, non solo riceverai un adeguato compenso, ma il tuo nome verrà pubblicato sulla rivista nello speciale elenco dei supercampioni; inoltre verrà pubblicato in netta evidenza, gratis e nelle stesse pagine, un qualsiasi tuo annuncio riguardante lo scambio di software: puoi facilmente capire il vantaggio che avrai rispetto a tutti gli altri annunci "normali" che vengono sempre stampati in altra parte della rivista.

ANCHE SE NON SEI BRAVO A GIOCARE

Non ti piace giocare, ma preferisci divertirti sproteggendo, manipolando i programmi, modificando o sostituendo la musica o i caratteri?
Anche in questo caso Commodore Computer Club ti offre l'opportunità di far conoscere a tutti la tua bravura.
Se possiedi la confezione originale di un qualsiasi software commercializzato (e solo in questo caso), e sei riuscito a rimuovere la protezione (e NON semplicemente a copiarlo con uno dei tantissimi copiatori in circolazione) mandaci una lettera descrivendo, in maniera semplice e chiara, il modo in cui ci sei riuscito.
Se, inoltre, sei riscuito ad "estrarre" la mappa dei **caratteri**, gli **sprite**, le varie schermate grafiche oppure la **musica** presente in un qualsiasi software commercializzato (originale o piratato), invia (su disco o cassetta) il file che sei riuscito ad estrarre insieme ad un breve **programma** (in Basic o in linguaggio macchina) che consenta di visualizzare la schermata, di far ascoltare la musica, di usare i caratteri ridefiniti.
Anche in questo caso riceverai un adeguato compenso ed il tuo nome apparirà nello speciale elenco degli utilizzatori "avanzati" dei computer Commodore ed avrai diritto alla pubblicazione di un annuncio speciale che consentirà di far conoscere il tuo nome a decine di migliaia di utenti italiani.
Per ottenere maggiori informazioni telefona al numero...
02 / 55500310 (lunedì e giovedì, ore 16:00 - 18:00)
...oppure invia il materiale richiesto a:
Commodore Computer Club
"Campioni del Software"
Via Mosè, 22
20090 Opera (Mi)

Si chiama Abacus. Si tiene dal **21 al 29 Aprile** nel **Padiglione 14** della **Grande Fiera d'Aprile**. Lí vi aspettano i computer (con programmi e periferiche) per i giochi intelligenti, per imparare una lingua, per fare musica, per disegnare, per scrivere la tesi di laurea, per giocare al totocalcio. E poi i lettori CD ROM per le enciclopedie, i telefoni più o meno intelligenti, i terminali videotext, i nuovi prodotti che "telematizzano" la casa. Con la possibilità di vederli, toccarli con mano e – in molti casi – acquistarli.

SYSTEMS INFORMA

Alcune informazioni saranno sicuramente apprezzate dai nostri lettori. Si tratta delle istruzioni dei *Gialli Commodore*, i cui programmi erano allegati nei fascicoli precedenti di *Commodore Computer Club*, e della novità telematica di casa Systems.

I GIALLI COMMODORE

Alcuni lettori hanno trovato in omaggio, nei precedenti fascicoli di Commodore Computer Club, una cassetta della serie *I Gialli Commodore* (formato C/64-128).

Per motivi di confenzionamento, i relativi manualetti contenenti le istruzioni non risultavano allegati al fascicolo in questione; nonostante ciò siamo sicuri che sono numerosi gli utenti che sono riusciti, egualmente, a giungere alla soluzione degli enigmi polizieschi.

Per coloro che, invece, hanno deciso di attendere ulteriori ragguagli, ecco di seguito alcuni preziosi consigli.

L'investigatore, per risolvere l'enigma, deve operare scelte logiche suggerite dagli stessi avvenimenti.

Per compiere azioni, spostamenti, interrogare, perquisire (ed altro) dovrà usare frasi composte da una, due, tre o più parole.

Per registrare su supporto magnetico la "situazione" raggiunta, in modo da riprenderla in un successivo momento, è sufficiente battere il comando *save*; tale opzione è assente in alcuni episodi *(Delitto a Villa Tsui)* per l'eccessiva semplicità degli stessi.

Per rivedere una pagina di testo, dopo una serie di *Input*, bisogna digitare il comando *text*.

Per visualizzare l'elenco dei protagonisti è necessario il comando *pers*.

Durante la fase delle indagini, se ritenete di averle concluse, potete comunicarlo per giungere alla fase finale in un qualsiasi Input; esempio: *Ho concluso le indagini*.

E' bene sottolineare che l'investigatore parla ed agisce in prima persona, anche sul piano linguistico.

In alcuni casi, riferendosi a persone nominate per nome o per cognome, digitare i loro nominativi con l'iniziale maiuscola (esempio: **H**elene e non **h**elene).

Riferendosi a persone morte (ad esempio, per una perquisizione), non si otterrà alcun effetto utilizzando il loro nome, ma solo parlando di *cadavere* oppure *corpo*.

UNA BBS TUTTA PER VOI

Ebbene sì: anche *Commodore Computer Club* mette a disposizione dei suoi lettori una **BBS**, attiva 24 ore su 24.

Per usufruire del servizio è necessario possedere un computer Commodore (C/64, C/128, Amiga, Ms-Dos compatibili) un modem, un idoneo programma di telecomunicazione, i programmi (de)compattatori più diffusi (Pkzip, Arc) e, nel vostro esclusivo interesse, un orologio a portata di mano per non superare i limiti messi a disposizione del vostro... portafoglio.

In questo momento, mentre questo articolo viene scritto, vige la decisone di offrire il servizio gratuitamente. In seguito sarà necessario, per accedere alla banca dati, digitare la parola d'ordine che, mese dopo mese, verrà pubblicata su *Commodore Computer Club*.

CHE COSA OFFRE LA BBS

Anzitutto, gli articoli, i programmi ed i files in genere, pubblicati sulla nostra rivista. La BBS sarà inoltre abilitata a ricevere lettere, articoli e programmi dei collaboratori.

Dovrebbe quindi tramontare, un po' per volta, l'epoca pionieristica che ci costringeva ad usare il borbonico (dis)servizio postale. D'ora in poi, trascorso l'inevitabile periodo di rodaggio, ogni comunicazione tra utenti e *Systems Editoriale* dovrà svolgersi via modem.

Naturalmente nessuno fa niente per niente ed i lettori ci perdoneranno se, in futuro, verranno prese le decisioni opportune per ovviare alla presumibile diminuzione di richieste di arretrati, *Directory, Amigazzetta* ed altri più o meno modesti introiti finanziari: il contenuto di articoli e dischetti dovrebbe, infatti, essere a disposizione "gratuitamente" via modem.

Forse in seguito il servizio verrà offerto ai soli abbonati (ai quali verrà comunicata la *password*) oppure questi avranno libero accesso a directory riservate.

L'importante è, per ora, iniziare.

Chi non l'ha ancora fatto, pertanto, provveda a procurarsi un bel modem, meglio se dotato di velocità minima di 1200 baud; se, poi, può viaggiare a 2400, tanto meglio per il vostro portafogli.

Il numero di telefono della BBS è:
02/ 52.49.211

Questo è lo stesso numero che, in precedenza, era indicato sulla nostra rivista come corrispondente alla Redazione di CCC.

Da questo momento, quindi, un tentativo di collegamento "vocale" con il numero indicato porterà solo un'emissione di fischi di varia intensità: è il computer che tenta di mettersi in contatto con voi!

Per le comunicazioni "normali", quindi, è opportuno comporre gli altri numeri di telefono della Systems Editoriale.

E' ovvio che, da questo momento, verranno priviliegiate le comunicazioni effettuate via modem - BBS.

VELOCITA', LINGUAGGI E COMPUTER

La lettera inviata da un nostro lettore è preziosa per ribadire concetti forse trascurati

di Alessandro de Simone

Purtroppo anche l'informatica va soggetta alle mode. In campo tecnico, infatti, dovrebbe essere un controsenso imitare un atteggiamento per il solo fatto che alcuni (forse solo più furbi di altri) decidono un modo di vestire, di comportarsi, di agire.

Se, però, nel campo dell'abbigliamento o delle auto è praticamente impossibile stabilire se un vestito è bello o brutto e se una macchina è più adatta ad uscire da una piscina piuttosto che a recarsi in Australia per vedere se ci sono ancora i canguri, nel campo dell'informatica c'è poco da scherzare: un bit è sempre un bit e la velocità di un'elaborazione può esser confrontata, orologio alla mano, con altre procedure "concorrenziali".

Ora è di moda il *Pascal* (pardon, il Turbo Pascal originale Borland) e la sera bande di giovanotti *Pascal*iani vanno a caccia di *Basic*hiani per fare a botte; a meno che, ovviamente, non invadano il territorio dei "*C*"iani; in quel caso scoccano scintille e scorre sangue.

Spesso, in redazione, sono costretto ad intervenire per sedare risse tra componenti di opposte fazioni che, con il passar del tempo, prendono coscienza di sè e (dal momento che siamo in Italia) provvedono prontamente a fondare un partito non appena si accorgono di essere almeno in due a condividere certe idee.

C'è infatti il partito di Amiga (le cui correnti interne, ferocemente divise in *Assembl*isti e *C*isti, trovano insperata e reciproca solidarietà quando giunge il momento di disprezzare gli *A-Basic*i); il partito Ms-Dos (che, a sua volta diviso in Cisti ed Assemblisti, è ormai prossimo alla scissione tra 386-isti e 486-isti; gli 8088 / 8086-isti sono stati fucilati tutti in ottobre).

I 64-isti compensano il recente sfoltimento delle loro schiere (dovuto al "tradimento" di quelli passati ad Amiga ed Ms-Dos) rafforzando la propria fede incrollabile nel popolare computer che, sostengono nel loro libro di preghiere, "fa tutto quello che riescono a fare un Amiga ed un Ms-Dos messi insieme" e, per convincersi, lanciano pubbliche condanne a morte eseguibili da chiunque sia in grado di catturare un Amighista sulla faccia della terra.

Attorno ai partiti gravita, come intuitivo, una maggioranza silenziosa e pagnottara che, a seconda dei casi, prende come riferimento politico un linguaggio (piuttosto che una macchina), un package (piuttosto che una procedura), una

CONVIENE COMPILARE?

Chi vi scrive è un ragazzo di 18 anni appassionato di computer e, ovviamente, di programmazione. Desidero esprimere la mia delusione per un prodotto Commodore (*Oxford Pascal*) ed ho ritenuto opportuno riferire le mie lamentele ad una rivista, come la vostra, che spesso, giustamente, esalta le macchine e gli accessori made in "CBM".

Il compilatore Pascal mi è stato regalato, con tanto di manuale in inglese . Ora le mie critiche non si puntano su "particolari" come la difficoltà d'uso, la lentezza della compilazione e le inevitabili restrizioni di comandi e di sintassi rispetto a dialetti ben più potenti e veloci come il *Turbo Pascal* della *Borland*, perchè tutti conosciamo i limiti hardware di un povero 8 bit quale il C/64.

Le mie critiche si puntano sul fatto che l'Oxford Pascal della Commodore, per lo meno la versione 1.0, risulta totalmente inutilizzabile per applicazioni serie che vadano al di là del piccolo esercizio di programmazione.

Mi è capitato, poco tempo fa, di scrivere in poche ore un programma in *Basic* sul calcolo statistico, a puro scopo di divertimento. Avendo avuto cura di compattarlo il più possibile, risultò lungo appena 8 blocchi, pur essendo abbastanza raffinato e completo. Decisi poi di compilarlo, per ottenere un codice più veloce, con *Austrospeed*. Risultato: un file compilato di 20 blocchi ed una velocità di quattro volte superiore.

A questo punto, soddisfatto della mia "creatura", ebbi la malaugurata idea di riscriverlo completamente con il famigerato Oxford, ottenendo un file sorgente di ben 21 blocchi (accidenti...) e neppure troppo veloce. Per evitare l'interminabile compilazione, oltre al caricamento del programma Oxford stesso, decisi di generare un file *Prg* eseguibile da Basic con il comando *Locate*, pensando che il passaggio da *P-code* a *linguaggio macchina* vero e proprio avrebbe giovato anche in termini di velocità.

Il risultato è stato disastroso: il file oggetto occupa 61 blocchi (quasi otto volte il file Basic normale e tre volte il file Basic compilato!), gira alla stessa velocità del sorgente compilato normalmente e... ha un modo alquanto strano di gestire il tasto *Stop*.

Ogni volta che il programma viene interrotto, parte una routine in macchina che accede al disco per ricaricare la testata del programma (in pratica la Sys di partenza da Basic).

Se il drive è stato precedentemente spento o, premendo contemporaneamente *Run / stop* e *Restore*, si è interrotto il funzionamento della famigerata routine macchina suddetta, il programma deve essere caricato daccapo. Che ne dite?

A questo punto mi chiedo a che cosa possa servire realmente un simile linguaggio, a parte qualche semplice esercizio per entrare in dimistichezza con il fantastico linguaggio Pascal.

Chiedo, quindi, se non è il caso di lasciar perdere del tutto l'idea di adoperare specifici linguaggi compilatori e di accontentarsi del solito Basic V.2, magari compilando i vari programmi con il popolare Austrospeed.

(Andrea Cilio - Genova)

stampante (piuttosto che un computer o un programma che la faccia funzionare).

ALL'INIZIO ERA IL BIT

La confusione, a mio modesto parere (a proposito, sono un *Basico*, ma non ditelo in giro) risiede nel fatto che troppo spesso si dimentica di esaminare l'intera "faccenda" informatica da un altro punto di vista: la sua *utilità* (intesa nel più vasto significato del termine) per l'uomo (e, siamo generosi, anche per la donna).

Un computer deve "*servire*" a qualcosa. Se, poi, tale "servizio" viene reso in un modo o in un altro, non sempre è importante.

Conosco alcuni utilizzatori (che definisco, senza offesa, "primordiali") che non sanno nemmeno dell'esistenza dei bit ed utilizzano un solo programma: Easy Script. Sono in grado, cioè, di accendere (nell'ordine) monitor, drive, stampante, drive e C/64 con il solo dito indice (ma accendono gli apparecchi uno alla volta).

In seguito, dopo aver digitato *Load "*", 8, 1,* attendono pazientemente il caricamento del programma che, una volta lanciato, trasforma questi modesti padri di famiglia (e/o mariti esemplari) in iperveloci dattilografi in grado di formattare un qualsiasi testo secondo un qualsivoglia formato; scrivere lettere d'amore, circolari condominiali, proteste ai giornali ed intere trattazioni sulle abitudini pomeridiane degli Assiri; e scrivono il tutto, badate bene, con due soli *diti* fantozziani (gli indici, ancora loro).

Formattano un disco (magari senza sapere bene il perchè), cancellano e duplicano file, conoscono il significato delle varie segnalazioni di errore, ammirano compiaciuti il meraviglioso foglio di carta che fuoriesce stancamente dall'asmatica 803 e mostrano orgogliosi il risultato dei loro lavori a mogli distratte, figli irriconoscenti, editori poco interessati ed Assiri che già conoscono l'argometo trattato.

Dunque, a questi utilizzatori primordiali non interessa affatto sapere che un computer Ms-Dos è in grado di memorizzare una lettera in 12 secondi contro il minuto necessario con il C/64: non hanno fretta; nè si sconvolgono al pensiero che su un dischetto Ms-Dos possono memorizzare fino a 1200 indirizzi da "collegare" a circolari personalizzate: l'unica volta che capita di inviare circolari si verifica in occasione delle assemblee condominiali, in cui il citofono è sufficiente per radunare i rissosi inquilini.

Il fatto, poi, che i word processor Ms-Dos offrono l'opportunità di pilotare perfino una stampante laser postscript (cosa che il C/64 *forse* non può fare), non provvede a far passare notti insonni.

Insomma, il vero utente dovrebbe assomigliare all'utilizzatore primordiale, inteso in senso buono.

E veniamo al dunque: è meglio usare un linguaggio piuttosto che un altro?

Come avrete già immaginato, non c'è (nè ci può essere) una risposta univoca in proposito: se una risposta ci fosse, nessuno vedrebbe usare i linguaggi "sconfitti".

Bisogna ricordare che, in ogni caso, è il microprocessore, coadiuvato dai vari chip che costituisco l'architettuta del sistema, a prendere l'ultima parola.

Vediamo di banalizzare il concetto e riferiamoci al caso in cui si voglia visualizzare la parola "pippo" ricorrendo a vari linguaggi.

In Basic è sufficiente...

```
100 Print "pippo"
```

...mentre in Pascal...

```
begin
writeln ('pippo ');
end.
```

In entrambi i casi, verificando l'occupazione su disco, sembrerebbe che lo spazio richiesto sia quasi identico: una manciata di byte.

Ciò si ferifica, però, solo considerando il programma Basic ed il listato Pascal con suffisso .pas (editabile con un w/p, per intenderci).

Se però vogliamo usare il listato Pascal *compilato*, cioè "slegato" dai vari file che costituiscono il linguaggio (il file con suffisso *.com*, vale a dire quello autonomo e direttamente eseguibile) questo arriva ad occupare diverse migliaia di byte a causa dell'inevitabile inglobamento delle librerie necessarie al suo buon funzionamento. Si potrebbe concludere, quindi, che il Basic è quello vincente, almeno in termini di economia di byte.

Ciò sarebbe vero se non si considerasse che il programma, per funzionare, ha bisogno dell'interprete (Basic, appunto) e che nel Commodore 64 è sempre disponibile, anche se l'utente non se ne accorge. Il mini programma, dunque, si anima di vita propria solo in apparenza: a manovrarlo come una marionetta provvedono gli 8 K di Rom Basic che risiedono permanentemente nel C/64 e che a loro volta sfruttano, dimenticavamo, varie altre dozzine di byte, poste prevalentemente in pagina zero, che sovrintendono alla gestione della zona del programma.

Tanto è vero che, cambiando computer, è indispensabile caricare prima l'interprete Basic e, da questo, caricare il programma. A pensarci bene, dunque, per stampare 'sto *Pippo* sono necessarie parecchie decine di Kbyte.

Per ciò che riguarda l'occupazione di memoria, dunque, sembrerebbe che l'ideale sia rappresentato dall'*Assembly* che, con una manciata di istruzioni, consente di visualizzare Pippo.

Poichè, però, nulla si crea e nulla si distrugge, da qualche parte, in memoria, devono pur esistere opportune routine in grado di visualizzare un gruppo di caratteri alfanumerci. Ed, infatti, se ci fate caso, in un qualsiasi programma l.m., in cui si verifichino tali necessità, il programmatore "salta" sempre ad alcune routine già presenti in *Rom* e che, guarda caso, appartengono ancora alle Rom del *Basic* se non del *Sistema Operativo.*

VARI CASI

Allora, riepiloghiamo:

Un qualsiasi programma deve, per forza di cose, prima o poi interagire con l'hardware della macchina.

Se un messaggio deve apparire sul video, deve passare attraverso il chip che gestisce il raster.

Ma per governare il chip bisogna attivarlo mediante routine opportune; ma per attivare le routine è necessario che queste siano presenti (non importa se su Rom oppure Ram) e chiamate in causa al momento giusto; prima di attivarle, però, è necessario che il messaggio stesso sia presente in vari byte successivi; ma per depositare i caratteri alfanumerici del messaggio è necessario che un "qualche cosa" li prelevi (da tastiera, da un file su disco o dall'interno di un programma).

SYSTEMS EDITORIALE PER TE

La voce III
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far *parlare* il computer, sia di farlo *Cantare!* Diversi esempi allegati.
Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.
Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.
Cassetta: L. 12000
Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).
Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese ed i guadagni di una famiglia.
Cassetta: L. 12000
Disco: L. 12000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.
Cassetta: L. 12000
Disco: L. 12000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.
Cassetta: L. 20000
Disco: L. 20000

Analisi di Bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.
Cassetta: L. 20000
Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 ed i rudimenti di programmazione. Interattivo.
Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.
Cassetta: L. 12000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica: ideale per far avvicinare i bambini al calcolatore. Diversi esempi allegati.
Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.
Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.
Solo su disco: L. 25000

Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive. Ideale per studenti.
Disco: L. 25000

L.M. + Routine grafiche
Un fascicolo speciale (corredato di dischetto) suddiviso in due parti: corso completo di linguaggio macchina 6502 e implementazione di numerose routine che aggiungono al C/64 istruzioni Basic specifiche per la grafica, comprese quelle per disegnare in prospettiva!
Fascicolo + disco: L. 16000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.
Disco: L. 15000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.
Disco: L. 15000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.
Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su *"Commodore Computer Club"*. In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro. Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.
Ogni dischetto: L. 12.000

AVVISO PER I LETTORI

La spiegazione relativa alla cassetta

I GIALLI COMMODORE

che alcuni lettori potranno trovare allegata a questo numero, si trova a pag. 10

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/0 sia in formato Microsoft sia in vesione C/64 (da chiedere, a parte, su disco). ***(125 pag.)***

L. 7000

Dal Registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. ***(94 Pag.)***

L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. ***(112 pag.)***

L. 5000

Utilities e giochi didattici
Raccolta di numerosi programmi, in versione C/64 e Spectrum, di particolare interesse per chi intenda svliluppare software didattico. ***(127 pag.)***

L. 6500

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. *(127 pag.)*

L. 7000

Dizionario del Personal Computer
Raccolta dei termini più diffusi nel campo professionale; dizionario inglese - italiano. (Edizione ridotta). ***(96 pag.)***

L. 8000

Dizionario dell'Informatica
Dizionario inglese italiano di tutti i termini usati nell'informatica. (Edizione completa). ***(385 pag.)***

L. 20000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word Star, Samna, Multimate Advantage, Word 3. ***(79 pag.)***

L. 5000

Telefax
Volumetto divulgativo sull'importanza del Telefax e sulle sue modalità operative caratteristiche. ***(66 pag.)***

L. 5000

Come compilare un giornale aziendale in Azienda
I principali problemi per chi opera in ambiente DPT, affrontati e risolti con la massima chiarezza e semplicità. ***(80 pag.)***

L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza. ***(91 pag.)***

L. 5000

ABBONAMENTO

Commodore Computer Club

11 fascicoli: L. 50.000

ARRETRATI

Ciascun numero arretrato

di C.C.C. L. 5000

COME RICHIEDERE I PRODOTTI SYSTEMS

Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.

Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale
Viale Famagosta, 75
20142 MILANO

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento") non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.

Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.

Volendo una spedizione in contrassegno è necessario anticipare la cifra di L. 10000 (diecimila), da inviare secondo le modalità prima indicate, indipendentemente dalla quantità di materiale richiesto, e da conteggiare, comunque, IN AGGIUNTA alla cifra risultante dall'ordine. (Si sconsiglia, pertanto, la richiesta di di prodotti in contrassegno)

Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

Systems Editoriale
Milano

Il *vero* problema, quindi, è quello di limitare al massimo i passaggi intermedi, sfruttando tutti gli accorgimenti possibili per eliminare perdite di tempo, evitare creazione di routine magari già disponibili all'interno della macchina, bypassare controlli forse inutili, modificare percorsi superflui.

Ecco perchè un buon programma in Basic risulta spesso più veloce ed efficiente di un listato in Pascal.

Se, poi, dopo aver ottimizzato un programma Basic, lo si compila, ecco che eventuali pecche intrinseche dei linguaggi interpreti possono essere limitate o, addirittura, annullate anche paragonando i risultati con veri linguaggi compilatori (C e Pascal, ad esempio).

UN CASO PRATICO

Riferiamoci, ora, ad un esempio volutamente banale che chiama in causa il Basic del C/64 ed un programma compilatore, il popolare *Austrospeed.*

Il listato che segue consente di effettuare alcune semplici considerazioni sulla velocità operativa ottenibile con il C/64.

```
100 rem prova di velocità
110 :
120 rem non compilato
130 rem righe 210-230: 44 sec.
140 rem righe 260-280: 25 sec.
150 ::
160 rem compilato (austrospeed)
170 rem righe 210-230: 39 sec.
180 rem righe 260-280: 24 sec.
190 :
200 gosub 300
210 for i=1 to 300
220 a = 3.14 * sqr (5 ^ 3) - log (56)
230 next: print ti$
240 x = 3.14: y = 5 ^ 3: z = 56
250 w = 1 :k = 300: gosub 300
260 for i=w to k
270 a = x *sqr(y)-log(z)
280 next: print ti$
290 end
300 ti$ = "000000": return
```

Tralasciando, per ora, i commenti riportati nelle prime righe, confrontiamo tra loro il gruppo di righe 200 - 230 e 240 - 280.

Entrambi i segmenti effettuano, in ultima analisi, le stesse identiche operazioni. Per trecento volte di seguito alla variabile numerica *A* viene associato il risultato di una operazione matematica.

L'attento lettore avrà sicuramente capito che si tratta di far "perdere" un po' di tempo all'elaboratore che, non potendo sapere che i calcoli sono sempre gli stessi, li esegue fedelmente per 300 volte.

Nel primo gruppo di righe, però, l'interprete Basic è costretto a "leggere" i quattro valori (3.14, 5, 3, 56) presenti nel listato come caratteri *Ascii*, "tradurli" nel formato opportuno e, finalmente, elaborarli; ciò si verifica anche per controllare se si è giunti al termine del ciclo *For... Next.*

Nel secondo gruppo di righe 260 - 280, invece, i quattro valori sono stati precedentemente associati ad altrettante variabili (x, y, z) e la "traduzione" prima vista non è più necessaria.

Lo confermano i risultati visualizzati: 44 secondi contro 25.

Chi conosce tale trucchetto, e lo applica costantemente in programmi che richiedono un elevato numero di elaborazioni matematiche, riesce a limitare i tempi di elaborazione.

"Se noi - starete pensando - addirittura compilassimo il listato con Austrospeed, dovremmo ottenere un ulteriore incremento di velocità".

La delusione, ovviamente, non tarda a venire. Compilando il programma (che richiede 25 blocchi contro gli appena due del Basic) ci accorgiamo che la velocità aumenta per ciò che riguarda il primo blocco di righe, ma rimane pressochè invariata nel secondo.

Probabilmente Austrospeed, durante la compilazione, sfrutta una procedura simile al trucchetto prima descritto e, nel caso in cui si sia già pensato ad ottimizzare, di più non può certo fare!

A PROPOSITO DI AMIGA

Se pensiamo al Basic di Amiga, poi, a nessuno sarà certamente sfuggita l'esasperante lentezza con cui un listato viene visualizzato. E' probabile che l'interprete necessiti ancora di una messa a punto, ma è più probabile che venga attivata una procedura complessa, costretta sempre a tener conto delle finestre aperte, dei task in attività, delle precedenze e di tante altre cose che, in un computer complesso come Amiga, sono piuttosto lunghe da gestire, perfino da un 68000.

Se, però, guardiamo con quale velocità l'istruzione *Fill* viene eseguita (una vera scheggia, per nulla confrontabile a quella di un interprete basic in ambiente Ms-Dos) potremmo erroneamente concludere che il basic di Amiga è velocissimo: il merito, al contrario, è tutto dell'hardware che, grazie ad una minima dose di istruzioni, è in grado di fare cose strabilianti.

L'istruzione basic, infatti, si limita a "passare" un pugno di parametri ad un *chip* grafico che, in modo addirittura indipendente dal 68000, provvede a far apparire quanto desiderato.

CONCLUSIONI

Azzuffarsi per stabilire se un computer, o un linguaggio, è "migliore" di un altro, è tempo perso.

Pensate, piuttosto, a dedicarvi ad un solo linguaggio, qualunque esso sia, ed approfondite le varie tecniche di programmazione consentite.

In parole povere, non pensate che il Basic sia limitante e limitato: esistono compilatori sempre più raffinati e potenti che sono in grado di eseguire elaborazioni in tempi sempre più competitivi.

Un esempio? In ambiente Ms-Dos troviamo il Turbo Basic ed il Quick Basic, in rapida e continua evoluzione.

Gli assemblatori ed i "C", poi, si sprecano; per non parlare dei Pascal.

I linguaggi compilatori, un tempo piuttosto difficoltosi e "duri", stanno diventando via via più snelli, semplici e strutturabili in modo molto intuitivo, paragonabile al Basic.

I moderni interpreti- compilatori Basic, dall'altra parte, cercano di aggiungere nuove caratteristiche che, per ciò che riguarda praticità e velocità, sono certamente paragonabili ai tradizionali linguaggi compilatori (C e Pascal in prima linea).

A patto, ovviamente, di non voler ostinatamente fare sempre e tutto con il C/64: con questo elaboratore è possibile lavorare, dignitosamente, solo in Basic e in Assembly.

Per gli altri linguaggi, lasciate perdere a meno di non volersi limitare a "curiosare".

Oppure usate altri computer.

CAMPUS 64 / 128

Le numerose lettere di protesta giunte dagli affezionati lettori di *Commodore Computer Club* hanno portato a più miti consigli le alte sfere della *Systems Editoriale*, che non hanno potuto fare a meno di ascoltare il grido di dolore che si levava dai petti degli smanettoni.
Pur se il recente aumento delle vendite viene attribuito all'incremento delle pagine dedicate ai videogames, è stata presa la decisione di diminuire lo spazio ad essi dedicato, a tutto vantaggio di articoli più *tosti*, vanto della precedente attività di tutti i collaboratori della rivista.
Si ventila, addirittura, un probabile aumento del numero delle pagine, da dividere equamente tra C/64, C/128 ed Amiga.
Per il momento, quindi, non possiamo dire di più, se non augurarci che il "ritorno" alle origini venga apprezzato in modo adeguato.
Due soli articoli compongono, stavolta, la parte di Campus dedicata ai "piccoli" computer della Commodore.
Tale decisione è stata presa per ovviare alla carenza di argomenti di livello "alto" verificatasi negli ultimi numeri.

Alcuni linguaggi, Pascal in testa, consentono di assegnare, ad uno stesso nome di variabile, due valori diversi, a seconda del "momento" in cui vengono elaborate.
Se, ad esempio, alla variabile numerica **X** viene assegnato il valore 15, quest'ultimo rimane immutato finchè non viene esplicitamente modificato.
Se, pertanto, ad un certo momento si "salta" ad una subroutine e questa modifica il contenuto di X (esempio: X = 54), al "ritorno" dalla subroutine il valore di X non sarà più 15 ma 54.
La procedura descritta nell'articolo, invece, consente di utilizzare lo stesso nome di variabile e di assegnare due valori diversi.

Chi possiede il C/128, e vuole usarlo in modo 128, non può sfruttare al massimo la versatilità del proprio sistema 128 / 1541 dal momento che, in circolazione, si trovano numerosissimi turbo - disk per C/64, ma non per C/128.
I possessori di drive 1571, ovviamente, non hanno problemi di sorta, ma gli utenti dei 128 / 1541 (che sono la stragrande maggioranza) sono costretti ad accontentarsi di velocità di trasferimento ridotte perchè, di solito, i turbo per C/64 non sono compatibili per agire in ambiente 128.
Ecco, quindi, un buon programma scritto da un nostro "campione del software", un personaggio al quale rivolgersi per ottenere utility e programmi vari per il C/128.

LE AVVENTURE DI

PRIMO GIOVEDINI

by M. Mietta
B. De Toffoli

Gara di acrobazia (1° e 2° file)

VARIABILI "LOCALI" E VARIABILI "GLOBALI"; ANCHE IN BASIC

Una routine in linguaggio macchina, per C/64, consente di emulare la notevole versatilità offerta dalla programmazione strutturata propria di linguaggi più evoluti, come il Pascal

di Armando Sforzi

Nel digitare il "caricatore" in Basic si consiglia, come al solito, di prestare la massima attenzione; è presente, tuttavia, un checksum che evita gli errori più clamorosi

Il Basic è sicuramente il linguaggio per computer più diffuso nel mondo grazie alle sue specifiche di immediatezza e relativa semplicità gestionale che lo rendono particolarmente gradito all'utenza, specialmente quando si trova al primo impatto col mondo informatico.

Tuttavia, appena l'uso si fa continuo, emergono ben presto qua e là *nei* più o meno vistosi e più o meno gravi che il popolare linguaggio purtroppo accusa. Per presentare la nostra routine vogliamo trarre spunto, appunto, da uno dei principali difetti del Basic individuabile nella sua elementare gestione delle variabili.

Sappiamo tutti, infatti, che esse, una volta definite, non possono essere in alcun modo cancellate, nè esiste la possibilità di un loro utilizzo dinamico all' interno del programma.

VARIABILI LOCALI

La routine che proponiamo in queste pagine offre una interessante soluzione alternativa arricchendo anche il Basic della straordinaria possibilità di creare *variabili locali*. Il concetto suonerà certamente familiare a chi per esempio conosce il linguaggio *Pascal*. Esso, infatti, per favorire al massimo la strutturazione logica della programmazione permette l'uso indipendente, nei vari sottoprogrammi, di variabili (e non solo di variabili) aventi la stessa etichetta. Ad esempio se la variabile denominata *Pippo%* vale 4 nel programma principale, passando ad un sottoprogramma essa potrà essere utilizzata con il valore di, poniamo, 6, sicuri del fatto che, una volta tornati nella routine di partenza, Pippo% automaticamente varrà di nuovo 4.

Quanto descritto è esattamente ciò che la nostra utility può fare anche nel Basic sia operando con variabili intere, reali o stringa (ma *non* opera sulle matrici). Vediamo il suo uso nel dettaglio.

IL PROGRAMMA

Inutile dire che bisogna digitare il (non troppo lungo) listato e lanciarlo per consentire l'allocazione dei codici macchina. Se tutto è andato a buon fine, sarà disponibile un nuovo "comando", da lanciare con l'opportuna Sys. Anzitutto è necessario, in apertura di programma, lanciare la routine con:

Sys 49152

In seguito avremo a disposizione un nuovo comando:

Ldim var1, var2, var3 ...

Con esso si potranno creare tutte le variabili locali che occorreranno.

Dopo *Ldim* le variabili che non sono state ridefinite continueranno a mantenere il proprio valore, come le *variabili globali* nel Pascal, mentre le variabili che costituiscono parametro di *Ldim* perderanno il valore fino ad allora posseduto (che ovviamente verrà memorizzato) e potranno essere utilizzate ex-novo in un sottoprogramma usuale chiamato da *Gosub* e terminante con *Return*. Quando il programma giungerà a questa istruzione terminale, prima di tornare al livello logico precedente, tramite la nostra routine richiamerà i contenuti precedentemente attribuiti alle variabili e cancellerà dalla memoria le ultime assegnazioni.

Naturalmente, e questo è un altro punto di forza di *Local/Var*, si può nidificare un numero qualsivoglia di livelli logici del tipo *Gosub / Return* fino al massimo consentito dallo *Stack* di Sistema.

Una particolarità si manifesta nel caso si definisca, con *Ldim*, una variabile che non esisteva già nel livello precedente. In questa situazione si hanno due possibilità:

1. Se questa nuova variabile verrà definita all' inizio del gruppo dei parametri, cioè:

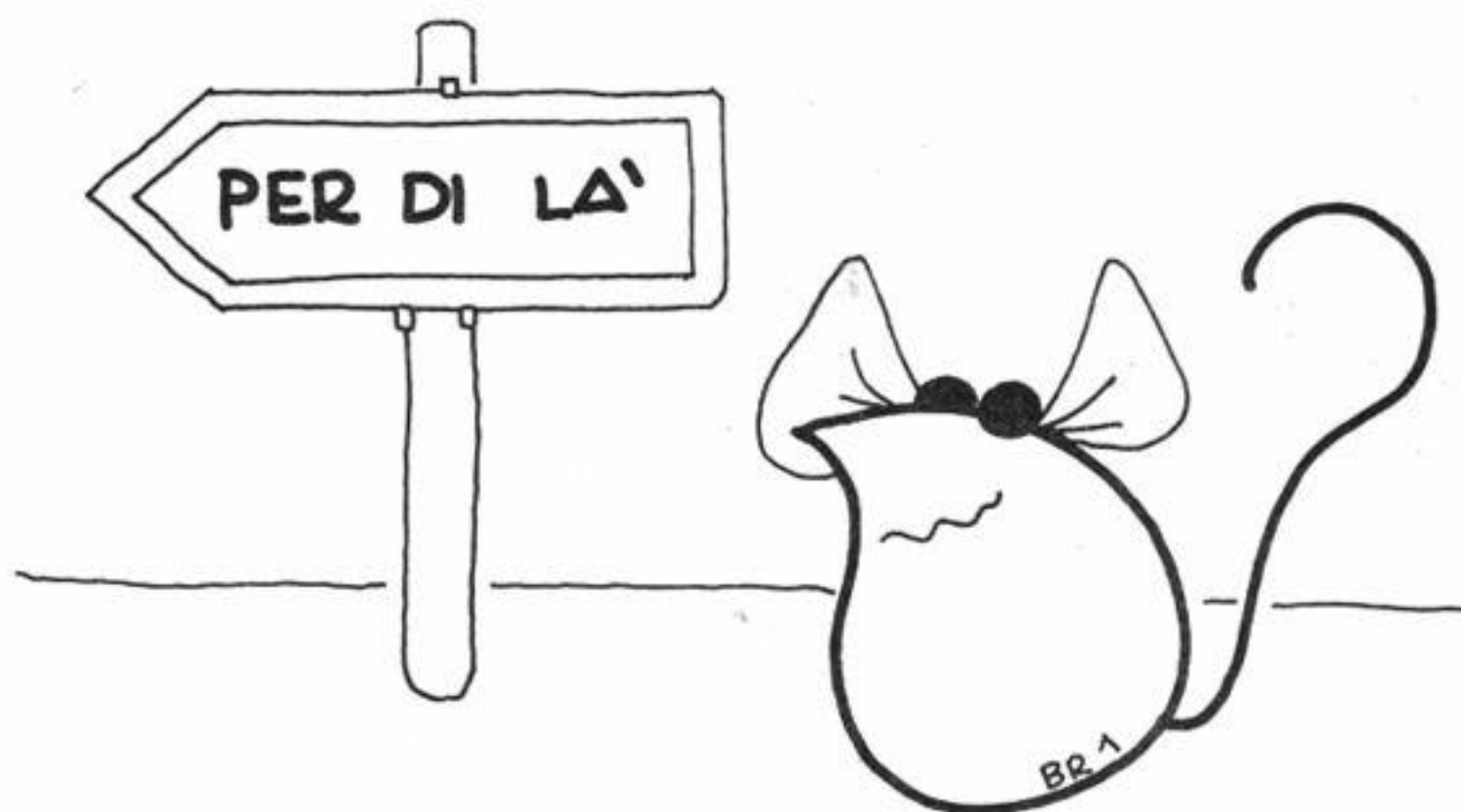
Ldim nuovavar, altrevar ...

...essa manterrà la propra definizione anche al ritorno dal sottoprogramma e ciò è utile quando si debbano "passare" dati da una subroutine al programma principale.

2. Se, invece, si desidera che la nuova variabile "muoia" col Return, basterà definirla in coda al gruppo dei parametri di Ldim:

Ldim altrevar, nuovavar

Il semplice demo allegato dovrebbe chiarire nella pratica l'utilizzo di questa singolare quanto preziosa utility, per mezzo della quale potremo finalmente creare programmi ben strutturati, anche lunghi e complessi, utilizzando un esiguo numero di variabili

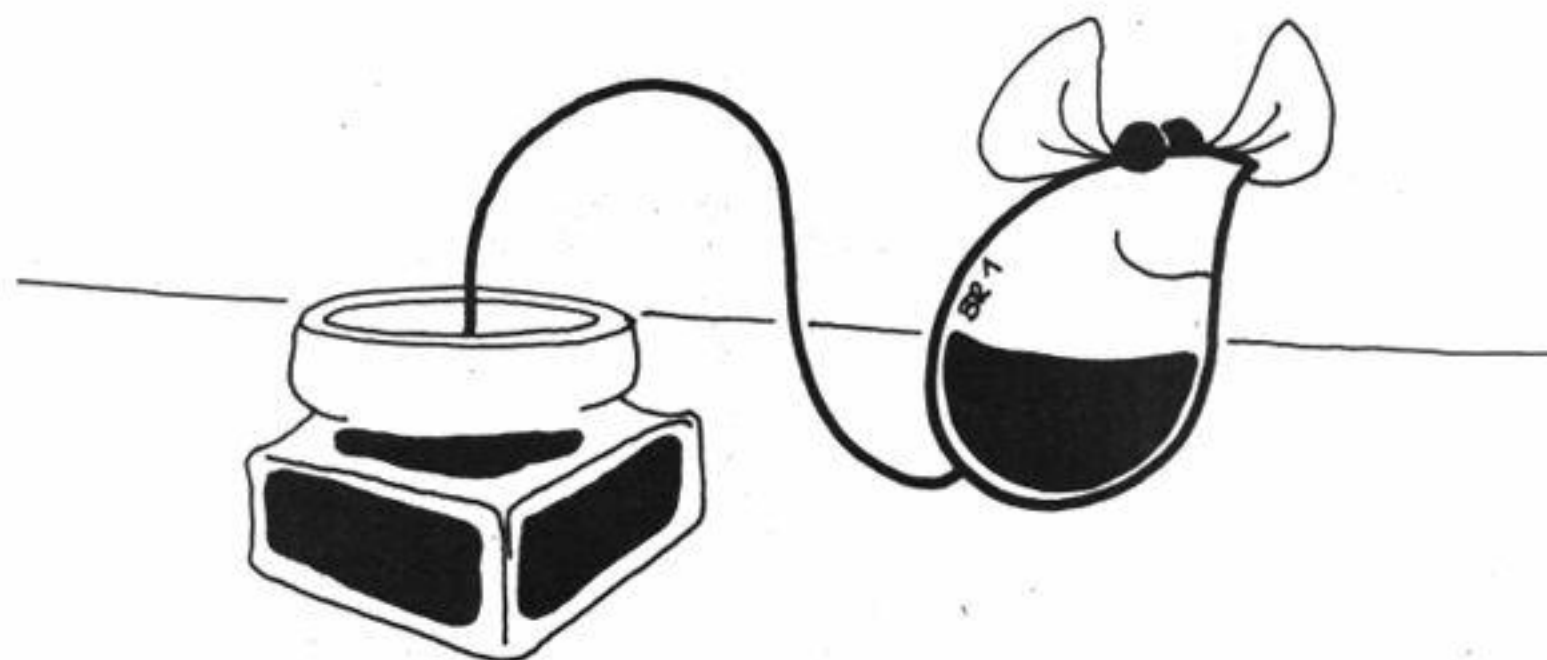
L'uso di variabili locali permette anche di sfruttare tecniche ricorsive che "liberano" l'utente dalla necessità di ricordare il nome con cui definisce le numerose variabili

```
0 REM  * LOCAL/VAR *  C/64
5 REM  BY ARMANDO SFORZI
7 REM         1990
9 :
10 S=0: FOR J=0 TO 373: READ A: POKE 49152 + J, A: S=S+A: NEXT
20 IF S<> 43002 THEN PRINT "ERRORE NEI DATA"
30 END
99 :
100 DATA 160,011,169,192,140,008,003,141,009,003,096,032,115,000,201
110 DATA 142,208,003,076,151,192,201,076,240,006,032,121,000,076,231
120 DATA 167,160,001,177,122,201,134,208,242,160,000,132,002,032,115
130 DATA 000,032,115,000,024,144,003,032,253,174,170,164,122,132,038
140 DATA 164,123,132,039,032,144,176,224,000,208,016,032,121,000,208
150 DATA 232,166,122,208,002,198,123,198,122,108,008,003,165,071,056
160 DATA 233,002,133,071,176,002,198,072,160,000,165,002,240,030,165
170 DATA 069,048,007,056,233,065,145,071,176,005,056,233,167,208,247
180 DATA 164,038,165,039,132,122,133,123,032,121,000,076,055,192,165
190 DATA 069,048,005,024,105,063,208,003,024,105,026,230,002,056,208
200 DATA 216,165,047,166,048,228,046,208,007,197,045,208,003,076,025
210 DATA 192,056,233,007,133,034,176,001,202,134,035,160,000,177,034
220 DATA 032,055,193,176,005,165,034,024,144,221,032,073,193,145,034
230 DATA 165,034,166,035,024,105,007,133,034,144,001,232,134,035,177
240 DATA 034,201,052,176,013,201,026,144,005,024,105,167,208,225,105
250 DATA 065,208,221,032,096,193,208,218,164,034,165,035,132,095,133
260 DATA 096,164,047,165,048,132,097,133,098,160,000,056,165,097,229
270 DATA 049,165,098,229,050,176,018,177,097,145,095,230,095,208,002
280 DATA 230,096,230,097,208,002,230,098,208,227,032,107,193,132,095
290 DATA 133,096,164,034,165,035,132,047,133,048,165,049,056,229,095
300 DATA 133,049,165,050,229,096,133,050,076,025,192,201,245,176,012
310 DATA 201,219,176,009,201,153,176,004,201,128,176,001,024,096,201
320 DATA 219,176,005,056,233,063,208,003,056,233,026,133,251,200,177
330 DATA 034,133,252,136,165,251,096,197,251,208,017,200,177,034,197
340 DATA 252,240,010,165,047,056,229,034,168,165,048,229,035,096
350 END
```

```
0 REM DEMO   LOCAL/VAR    C/64
10 PRINT CHR$(147) "DEMO LOCAL/VAR  C/64"
20 SYS 49152: REM ATTIVA 'LOCAL/VAR'
30 RV$=CHR$(18): A=1: B$="PIPPO": C%=2: D=10
40 PRINT RV$"CONTENUTO DELLE VARIABILI
50 PRINT RV$"DEFINITE NEL PROGRAMMA PRINCIPALE:"
60 PRINT "A="A: PRINT "B$="B$: PRINT "C%="C%: PRINT "D="D
70 :
80 GOSUB 190
90 PRINT RV$ "CONTENUTO DELLE VARIABILI"
100 PRINT RV$ "AL RITORNO DAL SOTTOPROGRAMMA"
110 PRINT "A=" A "         ← QUESTE HANNO RIPRESO..."
120 PRINT "B$=" B$ "    ← IL VALORE..."
130 PRINT "C%=" C% "       ← PRECENDENTE..."
140 PRINT "D=" D "    ← VARIABILE GLOBALE"
150 PRINT "E$=" E$ "   ← QUESTA E' RIMASTA DEFINITA"
160 PRINT "F$=" F$ "       ← QUESTA NON ESISTE PIU'"
170 END
180 :
190 LDIM E$, A, B$, C%, F$
200 A=5.7: B$="MILANO": C%=6: E$="MARE": F$="MONTE"
210 PRINT RV$ "VARIABILI NEL SOTTOPROGRAMMA"
220 PRINT "A="  A  "       ← VARIABILI..."
230 PRINT "B$=" B$ "   ← RIDEFINITE..."
240 PRINT "C%=" C% "      ← LOCALMENTE..."
250 PRINT "D="  D  "    ← VARIABILE GLOBALE"
260 PRINT "E$=" E$ "  ← NUOVA VARIABILE (LDIM E$,..)"
270 PRINT"F$="  F$ " ← NUOVA VARIABILE (LDIM ..,F$)"
280 RETURN
```

```
C/64: Disassemblato commentato di LOCAL/VAR

     Indirizzo di inizio: 49152 (C000)

. 0C000  A0 0B     LDY #$0B      ; Modifica...
. 0C002  A9 C0     LDA #$C0      ; il vettore...
. 0C004  8C 08 03  STY $0308     ; IGONE
. 0C007  8D 09 03  STA $0309     ;
. 0C00A  60        RTS           ;
. 0C00B  20 73 00  JSR $0073     ; Preleva un carattere
. 0C00E  C9 8E     CMP #$8E      ; E' il Token di 'RETURN'?
. 0C010  D0 03     BNE $C015     ;
. 0C012  4C 97 C0  JMP $C097     ; Se si', salta a SUB.RETURN
. 0C015  C9 4C     CMP #$4C      ; E' il carattere 'L'?
. 0C017  F0 06     BEQ $C01F     ; Se si', prosegue
. 0C019  20 79 00  JSR $0079     ;
. 0C01C  4C E7 A7  JMP $A7E7     ; Torna all' Interprete Basic
. 0C01F  A0 01     LDY #$01      ;
. 0C021  B1 7A     LDA ($7A),Y   ; Preleva il successivo carattere
. 0C023  C9 86     CMP #$86      ; E' il Token di 'DIM'?
. 0C025  D0 F2     BNE $C019     ; Se no, torna all' Interprete Basic
. 0C027  A0 00     LDY #$00      ;
. 0C029  84 02     STY $02       ;
. 0C02B  20 73 00  JSR $0073     ; Legge i parametri del comando
. 0C02E  20 73 00  JSR $0073     ;
. 0C031  18        CLC           ;
. 0C032  90 03     BCC $C037     ;
. 0C034  20 FD AE  JSR $AEFD     ;
. 0C037  AA        TAX           ;
. 0C038  A4 7A     LDY $7A       ; Memorizza...
. 0C03A  84 26     STY $26       ; la posizione corrente...
. 0C03C  A4 7B     LDY $7B       ; dei bytes di...
. 0C03E  84 27     STY $27       ; TXTPTR
. 0C040  20 90 B0  JSR $B090     ; Lancia il comando DIM
. 0C043  E0 00     CPX #$00      ;
. 0C045  D0 10     BNE $C057     ; Se la variabile c'e' gia' salta a MAIN
. 0C047  20 79 00  JSR $0079     ; Continua la routine di DIM...
. 0C04A  D0 E8     BNE $C034     ; fino al termine dei parametri
. 0C04C  A6 7A     LDX $7A       ;
. 0C04E  D0 02     BNE $C052     ;
. 0C050  C6 7B     DEC $7B       ;
. 0C052  C6 7A     DEC $7A       ;
. 0C054  6C 08 03  JMP ($0308)   ; Torna al Basic
. 0C057  A5 47     LDA $47       ;  ** MAIN **
. 0C059  38        SEC           ; Posiziona il vettore 71/72...
. 0C05A  E9 02     SBC #$02      ; sul primo byte...
. 0C05C  85 47     STA $47       ; del nome della variabile
. 0C05E  B0 02     BCS $C062     ;
. 0C060  C6 48     DEC $48       ;
. 0C062  A0 00     LDY #$00      ;
. 0C064  A5 02     LDA $02       ; E' la prima variabile di DIM?
. 0C066  F0 1E     BEQ $C086     ; Se si', salta a SUB.CRPVAR
. 0C068  A5 45     LDA $45       ;
. 0C06A  30 07     BMI $C073     ; Se la variabile e' reale o stringa
. 0C06C  38        SEC           ; sottrae 65 dalla prima lettera...
. 0C06D  E9 41     SBC #$41      ; del nome...
. 0C06F  91 47     STA ($47),Y   ; e memorizza il dato...
. 0C071  B0 05     BCS $C078     ;
. 0C073  38        SEC           ; Se la variabile e' intera...
```

```
. 0C074  E9 A7     SBC #$A7     ; sottrae 167 dalla prima lettera...
. 0C076  D0 F7     BNE $C06F    ; del nome e memorizza
. 0C078  A4 26     LDY $26      ;
. 0C07A  A5 27     LDA $27      ;
. 0C07C  84 7A     STY $7A      ; Il vettore TXTPTR riprende...
. 0C07E  85 7B     STA $7B      ; la vecchia posizione e...
. 0C080  20 79 00  JSR $0079    ;
. 0C083  4C 37 C0  JMP $C037    ; continua il comando DIM
. 0C086  A5 45     LDA $45      ; ** SUB.CRPVAR **
. 0C088  30 05     BMI $C08F    ;
. 0C08A  18        CLC          ; Se la variabile e' reale o stringa...
. 0C08B  69 3F     ADC #$3F     ; somma 63 alla prima lettera del nome
. 0C08D  D0 03     BNE $C092    ;
. 0C08F  18        CLC          ; Se la variabile e' intera...
. 0C090  69 1A     ADC #$1A     ; somma 26 alla prima lettera del nome
. 0C092  E6 02     INC $02      ; Incrementa il contatore delle variabili
. 0C094  38        SEC          ;
. 0C095  D0 D8     BNE $C06F    ; Torna a memorizzare i dati delle variabili
. 0C097  A5 2F     LDA $2F      ; ** SUB.RETURN **
. 0C099  A6 30     LDX $30      ; Se non ci sono variabili...
. 0C09B  E4 2E     CPX $2E      ;
. 0C09D  D0 07     BNE $C0A6    ;
. 0C09F  C5 2D     CMP $2D      ;
. 0C0A1  D0 03     BNE $C0A6    ;
. 0C0A3  4C 19 C0  JMP $C019    ; torna al Basic
. 0C0A6  38        SEC          ;
. 0C0A7  E9 07     SBC #$07     ;
. 0C0A9  85 22     STA $22      ; Carica il vettore per la ricerca...
. 0C0AB  B0 01     BCS $C0AE    ;
. 0C0AD  CA        DEX          ;
. 0C0AE  86 23     STX $23      ; delle variabili
. 0C0B0  A0 00     LDY #$00     ;
. 0C0B2  B1 22     LDA ($22),Y  ;
. 0C0B4  20 37 C1  JSR $C137    ; Cerca la prima variabile...
. 0C0B7  B0 05     BCS $C0BE    ; del livello logico precedente
. 0C0B9  A5 22     LDA $22      ; Se non la trova...
. 0C0BB  18        CLC          ;
. 0C0BC  90 DD     BCC $C09B    ; prosegue la ricerca
. 0C0BE  20 49 C1  JSR $C149    ; Va a ricostruire la prima variabile...
. 0C0C1  91 22     STA ($22),Y  ; e la memorizza
. 0C0C3  A5 22     LDA $22      ; Incrementa i puntatori...
. 0C0C5  A6 23     LDX $23      ; per ricostruire...
. 0C0C7  18        CLC          ; le successive variabili...
. 0C0C8  69 07     ADC #$07     ; se ci sono
. 0C0CA  85 22     STA $22      ;
. 0C0CC  85 01     BCC $C0CE    ;
. 0C0CE  E8        INX          ;
. 0C0CF  86 23     STX $23      ;
. 0C0D1  B1 22     LDA ($22),Y  ;
. 0C0D3  C9 34     CMP #$34     ;
. 0C0D5  B0 0D     BCS $C0E4    ; Ha incontrato una variabile non codificata
. 0C0D7  C9 1A     CMP #$1A     ;
. 0C0D9  90 05     BCC $C0E0    ;
. 0C0DB  18        CLC          ;
. 0C0DC  69 A7     ADC #$A7     ; Ricostruisce le variabili intere...
. 0C0DE  D0 E1     BNE $C0C1    ;
. 0C0E0  69 41     ADC #$41     ; reali e stringa
. 0C0E2  D0 DD     BNE $C0C1    ;
```

```
. 0C0E4   20 60 C1 JSR $C160      ; Controlla se sono terminate le variabili
. 0C0E7   D0 DA    BNE $C0C3      ; Se no, continua la ricerca
. 0C0E9   A4 22    LDY $22        ; Inizia una routine che termina a $0C136
. 0C0EB   A5 23    LDA $23        ; per spostare indietro i vettori...
. 0C0ED   84 5F    STY $5F        ; 47/48 e 49/50...
. 0C0EF   85 60    STA $60        ; per cancellare dalla memoria...
. 0C0F1   A4 2F    LDY $2F        ; il gruppo di variabili locali...
. 0C0F3   A5 30    LDA $30        ; che non serve piu'
. 0C0F5   84 61    STY $61        ;
. 0C0F7   85 62    STA $62        ;
. 0C0F9   A0 00    LDY #$00       ;
. 0C0FB   38       SEC            ;
. 0C0FC   A5 61    LDA $61        ;
. 0C0FE   E5 31    SBC $31        ;
. 0C100   A5 62    LDA $62        ;
. 0C102   E5 32    SBC $32        ;
. 0C104   B0 12    BCS $C118      ;
. 0C106   B1 61    LDA ($61),Y    ;
. 0C108   91 5F    STA ($5F),Y    ;
. 0C10A   E6 5F    INC $5F        ;
. 0C10C   D0 02    BNE $C110      ;
. 0C10E   E6 60    INC $60        ;
. 0C110   E6 61    INC $61        ;
. 0C112   D0 02    BNE $C116      ;
. 0C114   E6 62    INC $62        ;
. 0C116   D0 E3    BNE $C0FB      ;
. 0C118   20 6B C1 JSR $C16B      ;
. 0C11B   84 5F    STY $5F        ;
. 0C11D   85 60    STA $60        ;
. 0C11F   A4 22    LDY $22        ;
. 0C121   A5 23    LDA $23        ;
. 0C123   84 2F    STY $2F        ;
. 0C125   85 30    STA $30        ;
. 0C127   A5 31    LDA $31        ;
. 0C129   38       SEC            ;
. 0C12A   E5 5F    SBC $5F        ;
. 0C12C   85 31    STA $31        ;
. 0C12E   A5 32    LDA $32        ;
. 0C130   E5 60    SBC $60        ;
. 0C132   85 32    STA $32        ;
```

```
. 0C134  4C 19 C0 JMP $C019     ;
. 0C137  C9 F5    CMP #$F5      ; ** SUB.CNPVAR ***
. 0C139  B0 0C    BCS $C147     ; Controlla che la variabile...
. 0C13B  C9 DB    CMP #$DB      ; trovata sia la prima del gruppo
. 0C13D  B0 09    BCS $C148     ;
. 0C13F  C9 99    CMP #$99      ;
. 0C141  B0 04    BCS $C147     ;
. 0C143  C9 80    CMP #$80      ;
. 0C145  B0 01    BCS $C148     ; Carry=1: non e' la prima variabile
. 0C147  18       CLC           ; Carry=0: e' la prima variabile
. 0C148  60       RTS           ;
. 0C149  C9 DB    CMP #$DB      ; ** SUB.RCPVAR **
. 0C14B  B0 05    BCS $C152     ; Ricostruisce la prima variabile del gruppo
. 0C14D  38       SEC           ;
. 0C14E  E9 3F    SBC #$3F      ; Variabile reale o stringa
. 0C150  D0 03    BNE $C155     ;
. 0C152  38       SEC           ;
. 0C153  E9 1A    SBC #$1A      ; Variabile intera
. 0C155  85 FB    STA $FB       ; Memorizza in 251/252 il nome...
. 0C157  C8       INY           ; della prima variabile del gruppo
. 0C158  B1 22    LDA ($22),Y   ;
. 0C15A  85 FC    STA $FC       ;
. 0C15C  88       DEY           ;
. 0C15D  A5 FB    LDA $FB       ;
. 0C15F  60       RTS           ;
. 0C160  C5 FB    CMP $FB       ; ** Subroutine che controlla...
. 0C162  D0 11    BNE $C175     ; se e' terminata la ricerca...
. 0C164  C8       INY           ; delle variabili codificate..
.. 0C165 B1 22    LDA ($22),Y   ;
. 0C167  C5 FC    CMP $FC       ;
. 0C169  F0 0A    BEQ $C175     ;
. 0C16B  A5 2F    LDA $2F       ;
. 0C16D  38       SEC           ;
. 0C16E  E5 22    SBC $22       ;
. 0C170  A8       TAY           ;
. 0C171  A5 30    LDA $30       ;
. 0C173  E5 23    SBC $23       ;
. 0C175  60       RTS           ;
```

La sfida viene così formattata. Nei giorni seguenti, si procede ai preparativi per la competizione: l'incrociatore Aegis, selezionato quale giudice dell'incontro, comunica agli organizzatori la sua adesione...

C/128 & DRIVE1541: LANCIAMOLI A TUTTO GAS!

A grande richiesta, una indispensabile routine che manderà in visibilio gli smanettoni dotati di C128 & drive 1541.

Finalmente un eccezionale programma in grado di velocizzare il drive 1541 operando con il potente C/128

di Luca Viola

Chi possiede un C/128, dotato di un semplice drive 1541, spesso si rode il fegato durante gli estenuanti caricamenti a cui viene sottoposto, pensando con una punta di invidia ai più fortunati(?) possessori del drive 1570/71 (per non dire 1581!).

E pensare che per il solito modo 64 esistono decine di acceleratori software! Proprio non si può fare nulla per il nostro potente computer? Certo che si può!

Da un punto di vista strettamente tecnico, infatti, il modo di gestire i drive 1541 nelle due modalità è identico, eccezion fatta per la disposizione dei dati in memoria.

Nel C/128, infatti, si deve necessariamente tener conto del fatto che si hanno due banchi di Ram da 64K.

Tenendo conto di questa peculiarità, dovrebbe essere relativamente facile(!) scrivere un acceleratore software.

Ed in effetti e così.

LE CARATTERISTICHE

Il "turbo" che vedete in queste pagine è ricavato in misura diretta da quello pubblicato sul fascicolo *Commodore Speciale Drive* pubblicato dalla Systems Editoriale. In tale fascicolo, infatti, vi è il disassemblato commentato di un eccellente *turbodisk*. Operando una conversione diretta dalle routine kernal del C/64 al C/128 e convertendo alcune locazioni del S.O. del C/64 alle analoghe del C/128 ci si avvicina alla soluzione del problema.

I dolori nascono nel momento in cui si usano le routine di sistema per depositare in tutta la memoria disponibile i dati provenienti dal drive: tali routine, infatti, sono abbastanza lente, per cui la velocità globale, pur se soddisfacente, non sarebbe paragonabile a quella ottenibile con prodotti analoghi.

Si è dunque reso necessario riscrivere le suddette routine sostituendole con altre, più veloci ed efficienti, e il risultato finale si commenta da solo: per usare un termine di paragone comune tra i sessantaquattristi, lo *Speedisk 128* - questo è il suo nome - riesce a caricare, riferendosi a programmi registrati con un interleave pari a 10, **202 blocchi in soli 24 secondi!** E per chi era abituato a mangiar panini(!) durante le lungh(issime) pause di caricamento, è venuto il momento di mettersi a dieta...

Ma tralasciamo le facezie e passiamo ad analizzare più in dettaglio il programma.

Intanto, sulla portaerei vengono prese delle decisioni: l'incarico di comandante della pattuglia acrobatica viene affidato al nostro Primo Giovedini, il quale si reca nella sua cabina per riflettere sia sulla strategia da adottare, sia sull'uso del mouse al posto del joystick...

COME SI USA

Innanzitutto lubrificatevi i polpastrelli per digitare i circa *1200 Data* che costituiscono il programma caricatore in Basic. Per facilitare le cose il programma è diviso in blocchi, in modo da meglio individuare un eventuale errore. Una volta digitato il listato, salvatelo e verificatelo *prima* di dare run. A questo punto, se non avrete commesso errori, basterà premere il tasto Return per attivare il turbo quando il computer ve lo richiederà; diversamente apparirà una segnalazione di errore relativa al blocco di data che presenta problemi. Una volta attivato il turbo, potete registrarlo direttamente su disco come file Lm, digitando:

Bsave "Speedisk 128+", P 4864 To P 6076

Verrà così creato su disco un file contenente il nostro bravo turbo, che potrà essere caricato il memoria con un semplice:

Bload "Speedisk 128+"

...e attivato con:

Sys 5918

...se si vuole che il turbo sia immune al restore, oppure:

Sys 5888

...se si vuole caricare e attivare il turbo dall'interno di un proprio programma.

Una volta attivato, comunque, potrete caricare ciò che volete seguendo *la solita sintassi del basic 7.0* per i comandi di caricamento (e cioè *Load, Dload* e *Bload*), mantenendo inalterata, cioè, la procedura di caricamento di programmi sfruttando le routine standard del kernal.

Ciò significa che, se il turbo è attivato, una *JSR $FFD5* eseguirà un *Load Veloce*: naturalmente sarà necessario chiamare, nel modo consueto, le routine di *Setfile ($Ffba), Setname ($Ffbd)* e *Setbank* (*$Ff68,* richiede in *Acc.* il numero di banco in cui caricare) prima di saltare a *$Ffd5*. E' importante ricordare che la *verifica* sarà eseguita a velocità normale: si è scelta questa via per ottimizzare al meglio l'algoritmo di storaggio dati, tenendo anche conto del fatto che i programmi in genere vengono verificati una volta sola dopo il loro salvataggio, e quindi non si rende realmente necessaria l'operazione di verifica ad alta velocità (e poi, dite la verità, quante volte avete usato Verify o Dverify col drive?).

Se inoltre (caso raro) userete il comando *Load "$", 8* per caricare la directory, essa verrà caricata dalle routines standard del dos, perchè i turbodisk, in genere, maltrattano i dati provenienti dalla directory, non essendo questa un comune file programma. In ogni caso, per non sbagliare, potete continuare ad usare il comodo comando *Directory* (o *Catalog*). Caricando un programma, sullo schermo a 40 colonne apparirà un contablocchi in tempo reale nell'angolo in alto a destra, che vi informerà sul numero di blocchi (=gruppi di 255 bytes) che vengono deposti in memoria, mentre tale comoda(?) opzione non sarà visualizzata in 80 colonne. Sia in 40 che 80 colonne, invece, verranno visualizzati gli indirizzi di caricamento iniziale e finale del file trattato, ma solo se siete in modo diretto.

In modo programma, invece, l'informazione non verrà visualizzata per evitare di cancellare dati eventualmente presenti sul video, mentre il contablocchi sarà sempre visualizzato, quindi tenetene conto nei vostri programmi. Ricordate, infine, per evitare spiacevoli sorprese, di non invadere l'area compresa tra +4864 a +6106 (nel banco 0) e da $3e4 a $3f0 (comune a tutte le configurazioni di banco) perchè è qui che risiedono i dati vitali del turbo: in particolare, a partire da $3e4 si trova la miniroutine di storaggio dati in ram.

Il programma 2 è un demo che mostra visivamente l'incremento di velocità ottenuto: esso dapprima disegna una pagina grafica, quindi la registra su disco ricaricandola in seguito a velocità normale e, subito dopo, in turbo - e la differenza si nota!-

Il programma, ovviamente in linguaggio macchina, rimane residente in memoria e risulta "trasparente" ai comandi Basic di gestione dei file

"DENTRO" IL PROGRAMMA

Passiamo ora ad una breve descrizione delle varie sezioni del turbo (per il disassemblato si rimanda al fascicolo *Speciale Drive*, dal momento che i due programmi sono più o meno simili (le differenze più rilevanti risiedono, appunto, nella gestione della memoria, fondamentalmente diversa tra il C/64 e il C/128). I salti utilizzati, comunque, sono quelli standard del *Kernal Cbm*, e che quindi potrete facilmente individuare.

Le routine non comprese nella tabella di salto del kernal sono:

$F50F (stampa Searching For...)
$F533 (stampa Loading)
$F685 (stampa File Not Found)
$F0D5 (routine di Bopen: effettua una Open veloce saltando alcuni controlli)
$F59E (routine di Bclose: effettua una Close veloce
$Ff77 (carica il valore puntato dalla locazione di pagina zero posta in accumulatore + un offset posto in Y nel numero di banco posto nel registro X)

Vediamo ora in dettaglio le sezioni del programma:

$1300-$13Ff: questo programma va trasferito alla memoria del drive
$1400-$14Bd: procedura di apertura del file
$14c0-$1551: lettura e storaggio dati
$1552-$1700: chiusura del file o errore
da $1700 in poi è presente il codice di inizializzazione che modifica il vettore di load, stampa il messaggio iniziale, modifica il vettore di Restore, pone in $3e4 la routine di storaggio veloce, ecc.

In ogni caso, se conoscete il L.M., non dovreste aver problemi a comprendere, almeno nelle sue linee essenziali, il programma.

L'importante è che la schiavitù dei caricamenti lunghi *e' finita!!*

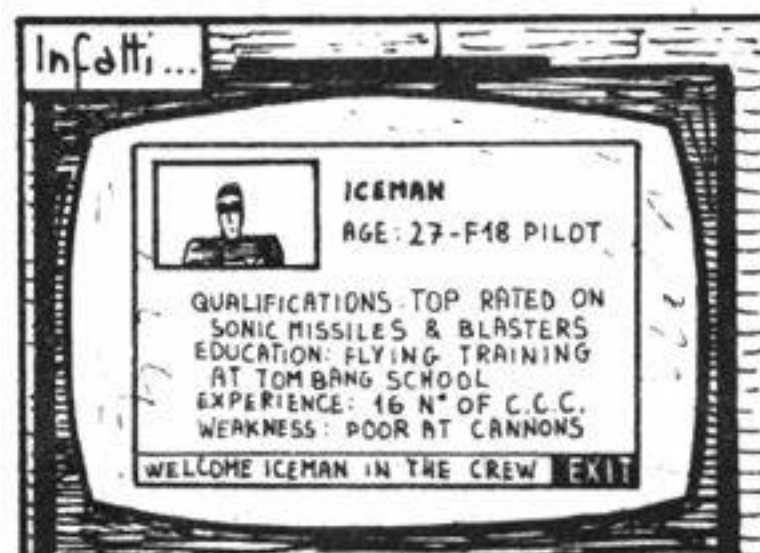
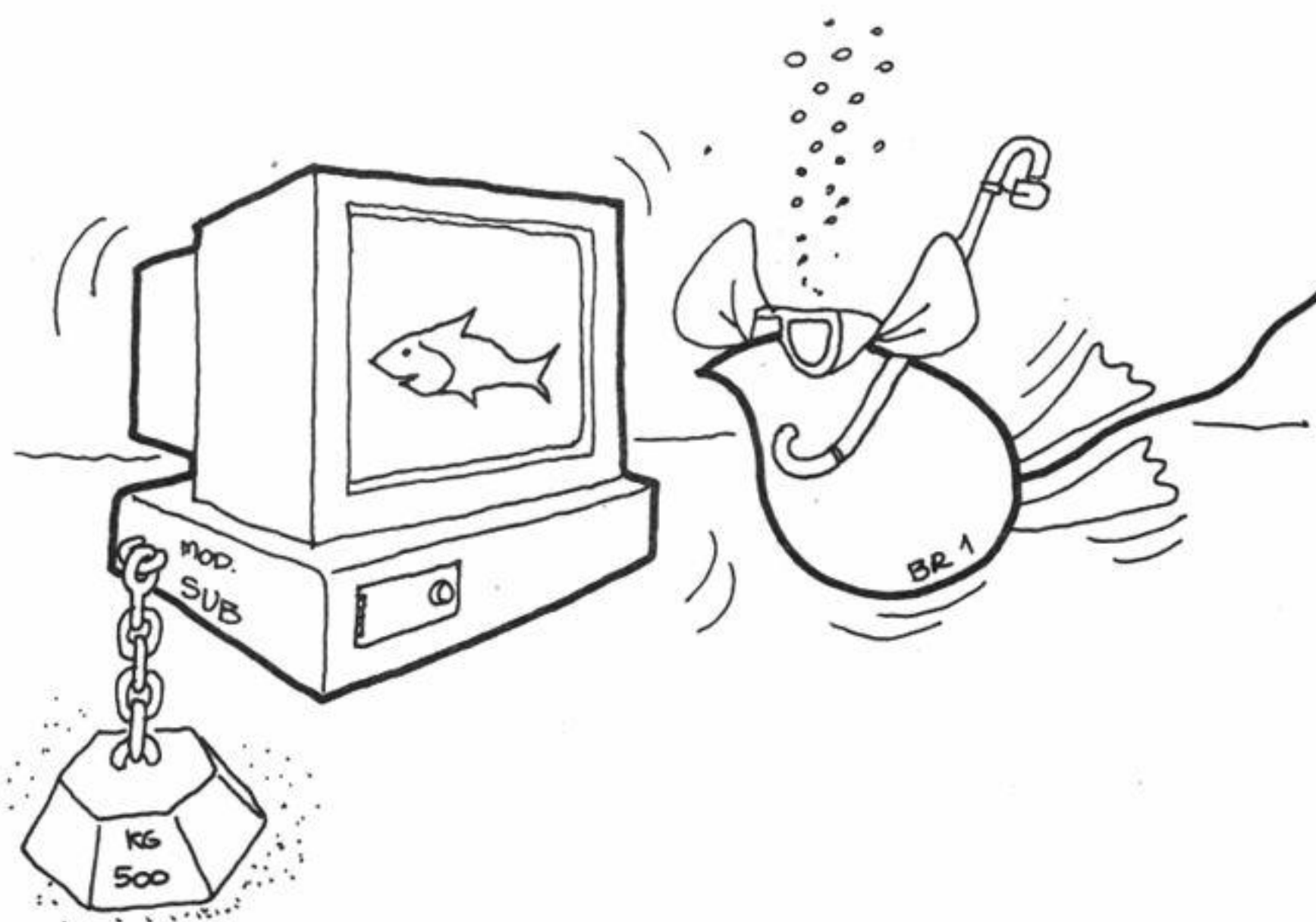
Il fascicolo "Speciale Drive", cui si fa cenno in queste pagine, può essere richiesto al nostro Servizio Arretrati

INTERLEAVE E DINTORNI

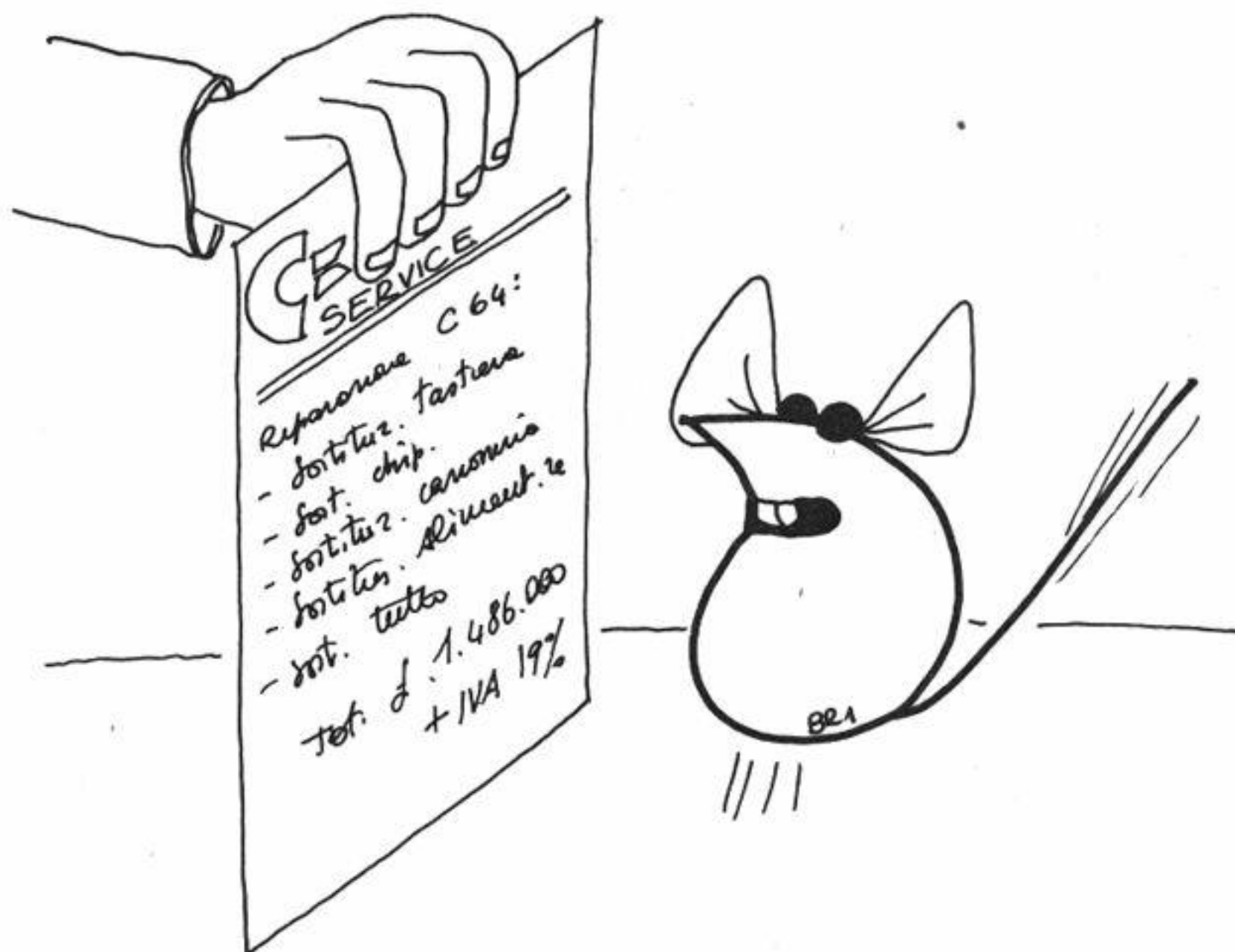
Per Interleave si intende il numero di settori dopo cui il Dos del drive scriverà, in fase di registrazione, il settore successivo.
Per esempio, con un *interleave 10*, se un programma viene registrato su una certa traccia a partire dal settore zero, il Dos sceglierebbe, come settore da utilizzare successivamente per la registrazione, il *decimo*.
Almeno intuitivamente, più tale valore è piccolo, più veloce è il caricamento di un file, perchè vengono eliminati i tempi morti per lo spostamento della testina.
Ma ciò, nella pratica, non si verifica in quanto ogni velocizzatore ha un suo interleave ideale, al di sotto o al di sopra del quale rallenta vistosamente (e questo si verifica soprattutto su dischetti troppo pieni o sottoposti a vari comandi di tipo Scratch & save, perchè non sempre il Dos riesce a trovare il giusto numero di settori liberi per le sue esigenze). L'interleave ideale per il nostro velocizzatore è pari a *9* o *10*, ed è possibile stabilire, purchè si abbia un dischetto sufficientemente libero, l'interleave con cui salvare i programmi. Se quindi, prima di registrare il vostro file, digitate:

```
Open 1, 8, 15, "M-W" + Chr$(105) + Chr$(0) + Chr$(1)
+ Chr$(in): Close 1: Dsave "Nome"
```

(dove *in* rappresenta l'interleave e può avere valore 9 o 10) otterrete il massimo rendimento dal turbodisk.

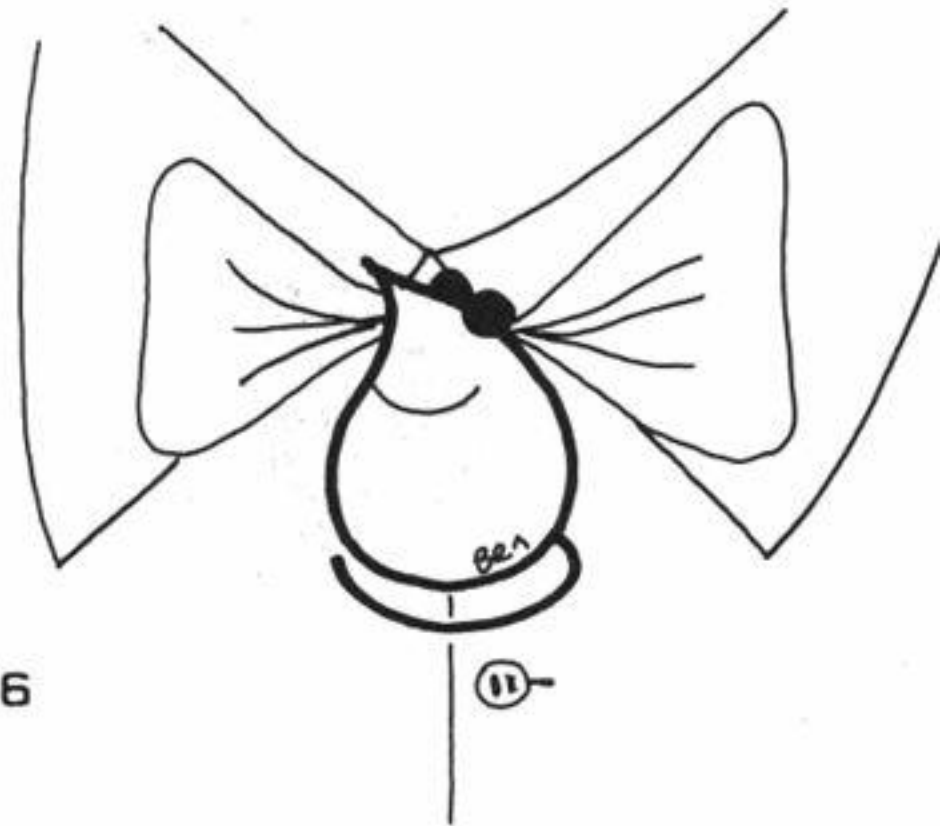
Per un maggiore approfondimento su questo argomento si rimanda comunque al 69 di CCC, pag. 2

```
10 rem turbo disk per c/128
20 :
100 scnclr:trap 400
110 color0,1:color 4,1:color6,1:color5,6:printchr$(14);chr$(11)
120 restore 1000:print"Attendere... sto leggendo i dati"
130 fast:lc=4864:bl=0:k=0:ck=0:sum=0:a=0:a$=""
140 do
160 read a$:if a$="*" or a$="**" then exit
170 a=dec(a$):poke lc+k,a:k=k+1
180 sum = sum +a
190 loop
200 bl=bl+1:read ck$:ck=dec(ck$):if ck<>sum then print "errore nel blocco #";bl:
slow:stop:else print"blocco #";bl" ok"
210 if a$ <> "**" then sum = 0 :goto 140
220 slow
230 print"*** Turbo disk per C128 + drive 1541 ***"
235 print:print"by VIOLA LUCA - (@) 1990"
240 print:print"Il turbo velocizza le operazioni di load"
250 print:print"mantenendo inalterata la sintassi del    "
260 print:print"BASIC 7.0. E' immune da STOP + RESTORE   "
280 print:print"ma viene disattivato dal reset.          "
290 print:print"Per riattivarlo : SYS 5918 <R>"
300 print:print:print"RETURN per attivarlo"
310 do:get a$:loop until a$=chr$(13):sys 5918
320 :
330 end
340 :
400 slow:print:print err$(er);" in";el:help:end
```

Il lettore attento (ma anche quello un po' meno attento, purchè dotato di una buona FLOW CHART) avrà sicuramente notato che compaiono sempre i soliti nomi in questo fumetto...

Ragazzi, scusate, ma il fatto è che devo aver perso il secondo PILOTS' NAME DATA DISK...

```
500 :
600 rem ***** blocco 1 ****
610 :
1000 data a9,06,85,31,20,0a,f5,50,fe,b8,ad,01,1c,91,30,c8,d0,f5,a0,ba,50,fe,b8
1010 data ad,01,1c,99,00,01,c8,d0,f4,20,e0,f8,a5,38,c5,47,f0,04,a9,04,d0,5f,20
1020 data e9,f5,c5,3a,f0,04,a9,05,d0,54,b1,30,d0,03,ee,01,06,b1,30,aa,2c,00,18
1030 data 10,fb,a9,10,8d,00,18,2c,00,18,30,fb,8a,4a,4a,4a,4a,8d,00,18,0a,29,0f
1040 data 8d,00,18,8a,29,0f,8d,00,18,0a,29,0f,8d,00,18,a2,0f,ea,8e,00,18,ad,00
1050 data 06,f0,1c,c8,d0,c6,a9,05,85,0c,ad,01,06,85,0f,b1,30,c5,0e,85,0e,f0,05
1060 data a9,01,4c,69,f9,4c,04,07,c8,cc,01,06,d0,a7,a9,7f,d0,f0,00,00,00,00,00
1070 data 00,00,00,00,00,00,00,00,00,00,a9,10,8d,05,1c,a5,18,85,0e,a5,19,85,0f
1080 data a9,05,85,0c,a9,e0,85,04,a5,04,30,fc,c9,7f,f0,30,c9,02,90,ec,c6,0c,10
1090 data ec,a4,0c,c8,d0,07,a9,c0,20,8e,d5,c6,0c,a4,0c,c0,fb,b0,da,a9,60,8d,71
1091 data *,5ac1
1092 :
1093 rem ***** blocco 2 *****
1094 :
1100 data 07,a9,ff,20,41,07,a9,3a,8d,05,1c,a5,04,a2,04,4c,0a,e6,a9,3a,8d,05,1c
1110 data 4c,48,d0,85,93,a5,ba,c9,04,b0,05,a5,93,4c,6c,f2,a5,b7,d0,01,60,20,cf
1120 data 17,ea,c9,24,f0,ee,a5,ba,85,b8,20,0f,f5,20,cc,ff,a6,b9,86,97,a9,60,85
1130 data b9,20,d5,f0,a5,ba,20,b4,ff,a5,b9,20,96,ff,20,a5,ff,a5,90,4a,4a,90,03
1140 data 4c,aa,17,20,33,f5,a9,30,8d,25,04,8d,26,04,8d,27,04,a9,00,85,a4,20,a0
1150 data 16,a9,57,20,a8,ff,a5,a4,20,a8,ff,a9,07,20,a8,ff,a9,20,20,a8,ff,a4,a4
```

```
1160 data 18,a5,a4,69,20,85,a4,b9,00,13,20,a8,ff,c8,c4,a4,d0,f5,20,ae,ff,a5,a4
1170 data d0,ce,20,a0,16,a9,45,20,a8,ff,a9,ab,20,a8,ff,a9,07,20,a8,ff,20,ae,ff
1180 data ad,00,dd,48,29,07,85,a5,09,08,85,a4,ad,15,d0,48,a9,00,8d,15,d0,a5,01
1190 data 48,78,a9,73,24,01,a2,fc,ea,ea,ea,a9,7f,8d,00,dc,cd,01,dc,f0,07,20,20
1191 data *,702d
1192 :
1193 rem ***** blocco 3 *****
1194 :
1200 data 16,c9,ff,d0,03,4c,52,15,48,20,20,16,a8,68,48,e0,fe,08,c9,00,d0,0a,98
1210 data aa,ca,ca,28,08,f0,02,ca,ca,28,f0,14,20,20,16,85,ae,20,20,16,a4,97,d0
1220 data 06,a5,c3,85,ae,a5,c4,85,af,20,20,16,20,e4,03,e6,ae,d0,02,e6,af,ca,d0
1230 data f1,ad,27,04,c9,39,f0,05,ee,27,04,d0,19,a9,30,8d,27,04,ad,26,04,c9,39
1240 data f0,05,ee,26,04,d0,08,a9,30,8d,26,04,ee,25,04,a2,fe,68,f0,29,4c,c0,14
1250 data ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,a9,02,85,90
1260 data 4c,03,40,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,ea,68,85,01,68,8d,15,d0,68
1270 data 8d,00,dd,20,9e,f5,a9,40,05,90,85,90,4a,4a,08,20,7e,17,28,20,b4,16,a9
1280 data 1d,58,60,a5,97,d0,07,a6,c3,a4,c4,b8,50,2f,20,a0,16,a9,52,20,a8,ff,a9
1290 data 02,20,a8,ff,a9,04,20,a8,ff,a9,02,20,a8,ff,20,ae,ff,a5,ba,20,b4,ff,a9
1291 data *,74c7
1292 :
1293 rem ***** blocco 4 *****
1294 :
1300 data 6f,20,96,ff,20,a5,ff,aa,20,a5,ff,a8,20,ab,ff,20,c7,15,20,b4,16,a9,20
1310 data 20,d2,ff,a9,24,20,d2,ff,98,20,d6,15,8a,48,29,f0,20,e1,15,68,29,0f,10
1320 data 04,4a,4a,4a,4a,a8,18,69,30,c0,0a,90,03,18,69,07,4c,d2,ff,ea,ea,ea,ea
1330 data ea,ea,ea,ea,ea,ea,ea,ea,ea,a0,a0,a0,a0,a0,a0,a0,50,50,50,50,50,50
1340 data 50,50,0a,0a,0a,0a,0a,0a,0a,0a,05,05,05,05,05,05,05,05,a5,a4,8d,00,dd
1350 data ad,00,dd,10,fb,ad,12,d0,c9,00,90,06,29,06,c9,02,f0,f3,a5,a5,8d,00,dd
1360 data 20,60,16,60,20,20,20,20,20,20,20,20,10,10,10,10,10,10,10,10,02,02,02
1370 data 02,02,02,02,02,01,01,01,01,01,01,01,01,ea,2e,3f,16,6e,3f,16,ac,00,dd
1380 data b9,00,16,ac,00,dd,19,08,16,ac,00,dd,19,10,16,ac,00,dd,19,18,16,60,80
1390 data 80,80,80,80,80,80,80,40,40,40,40,40,40,40,40,08,08,08,08,08,08,08,08
1391 data *,50b0
1392 :
1393 rem ***** blocco 5 ****
1394 :
1400 data 04,04,04,04,04,04,04,04,a5,ba,20,b1,ff,a9,6f,20,93,ff,a9,4d,20,a8,ff
1410 data a9,2d,4c,a8,ff,a6,ae,a4,af,60,ea,ea,ea,ea,ea,ea,00,00,00,00,00,00
1420 data 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
1430 data 00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00
1440 data 00,00,00,00,00,00,00,00,00,00,00,00,a9,74,8d,30,03,a9,17,8d,31,03,60
1450 data a5,c6,a6,c7,20,68,ff,a9,ae,8d,b9,02,a9,00,85,93,4c,00,14,a9,00,8d,20
1460 data d0,8d,21,d0,a9,0f,85,f1,20,9b,41,a9,07,85,f1,20,7d,ff,0d,53,50,45,45
1470 data 44,49,53,4b,20,31,32,38,20,2b,20,31,35,34,31,20,56,31,2e,30,0d,0d,42
1480 data 59,20,56,49,4f,4c,41,20,4c,55,43,41,0d,00,a9,6e,8d,00,0a,a9,17,8d,01
1490 data 0a,20,00,17,4c,03,40,20,1e,17,4c,03,40,c9,01,f0,03,4c,86,17,4c,6c,f2
1491 data *,3e76
1492 :
1493 rem ***** blocco 6 *****
1494 :
1500 data a5,7f,f0,01,60,4c,87,15,a2,00,bd,96,17,9d,e4,03,e8,e0,0d,d0,f5,4c,b0
1510 data 17,a0,00,8c,00,ff,a0,00,91,ae,8c,00,ff,60,ea,ea,ea,ea,ea,ea,ea,20,83
1520 data a1,4c,85,f6,ad,30,d0,48,ea,a4,c6,be,f0,f7,8e,e5,03,a9,00,8d,30,d0,20
1530 data 0b,17,68,8d,30,d0,a9,ae,8d,f0,03,60,a0,00,a9,bb,a2,01,20,74,ff,60,00
1531 data **,2fa2
1540 :
1550 end
1560 :
1570 rem" Turbo per C128 + Drive 1541
1580 rem"   (@) 1990 by Luca Viola
1590 rem"        Version 1.0+
```

Così, dopo che Primo ha stupito i compagni con l'ennesima citazione di un videogame che va per la maggiore, i piloti si apprestano al decollo.
Come andrà questa prima seduta di allenamento all'acrobazia, lo saprete fra poco, nel 2° file di questa storia ...

```
10 rem *** demo speedisk.***
20 rem *** necessita che ***
30 rem ***  lo speedisk  ***
40 rem *** sia attivato! ***
50 :
60 :
100 scnclr:scratch "#screen"
110 color0,1:color4,1:color1,16
120 graphic1,1
130 locate 160,100
140 for t=0 to 360
150 drawl to 85;t:locate 160,100
160 next
170 circle1,0,0,50:paint 1,0,0
180 circle1,319,0,50:paint1,319,0
190 circle1,0,199,50:paint1,0,199
200 circle1,319,199,50:paint1,319,199
210 draw1,15,15 to 305,15 to 305,185 to 15,185 to 15,15
220 paint 1,159,1:paint1,161,1:paint 1,319,100
222 paint1,159,199:paint1,161,199:paint1,1,100:
230 color0,1:color1,16
240 bsave "#screen",p 8192 to p 16192
250 graphic1,1
260 poke 816,108:poke 817,242:rem turbodisk off.
270 bload "#screen",p8192
280 sys 5888:rem turbodisk on!
290 scnclr :bload"#screen",p8192
300 sleep 3:graphic0:scratch"#screen"
310 end
320 end
330 :
340 rem ** attenzione: il programma **
350 rem ** crea e cancella un file   **
360 rem ** di nome "#screen". veri-  **
370 rem ** ficate che un tale file   **
380 rem ** non sia presente sul vo-  **
390 rem ** stro disco o che vi sia   **
400 rem ** spazio a sufficienza      **
410 :
420 prg by luca viola (@) 1990
```

(segue a pagina 65)

A SCUOLA DALLO STREGONE

Applichiamo al computer le magiche regole di una scienza inesatta: la cabala

di Gregor Samsa

Qualcuno, forse, si sarà accorto che da un po' di tempo a questa parte, anche nel tabellone luminoso del tabaccaio sotto casa comincia ad apparire la fatidica scitta "Ricevitoria Lotto n...".

Eh già, sembra che l'italianissimo sogno di tentare di arricchirsi, non passi più solo attraverso una schedina del Totocalcio. Di che cosa si sta parlando?

Semplice: di sondaggi di opinione. Pare, infatti, che il gioco del Lotto stia rivivendo una seconda giovinezza, uscendo dal suo tradizionale ambito.

Quello, per intenderci, della vecchiétta che, con voce tremolante, spiega al titolare di un botteghino, con estrema dovizia di particolari, il mirabolante sviluppo onirico dell'ultima notte.

Aspetti pittoreschi a parte, forse non tutti sanno che la cosiddetta *cabala*, per lo più identificata con l'interpretazione (non certo Freudiana) dei sogni, in realtà poggia su ferree basi matematiche.

Con ciò, si badi, non si intende affermare che sia una scienza esatta (la collaborazione con questa rivista potrebbe risentirne pesantemente!).

I calcoli, spesso complicati ed astrusi, hanno sempre un'origine decisamente meno logica.

Trattasi, comunque, pur sempre di calcoli, per giunta con procedure talvolta piuttosto lunghe da eseguire manualmente.

Ergo, l'ideale per manipolazioni computerecce.

Se vi state chiedendo perchè occuparsene, la risposta non è univoca: per nobili scopi didattici, per meno nobile e divertita curiosità, nonchè per decisamente ignobili fini di lucro.

Un terno al Lotto, in fondo, non ha mai fatto male ad alcuno.

UNA PIRAMIDE... INSOLITA

Ci occuperemo brevemente, in questa sede, di due tecniche cabalistiche che si propongono lo stesso esito: ottenere dei numeri "magici" partendo da nomi di persone, frasi dette o sognate, eccetera.

Senza, però, troppa fatica digitereccia.

Entrambe, infatti, sono applicate nel breve listato di queste pagine, strettamente basic, e sfruttabile senza alcuna modifica tanto sul **C/64** che sul **C/128** (modo 128).

```
PIRAMIDE MAGICA
criptogramma di: COMPUTERCLUB
6 8 8 6 0 9 3 4 6 0 0 5
 4 6 4 6 9 2 7 0 6 0 5
  0 0 0 5 1 9 7 6 6 5
   0 0 5 6 0 6 3 2 1
    0 5 1 6 6 9 5 3
     5 6 7 2 5 4 8
      1 3 9 7 9 2
       4 2 6 6 1
        6 8 2 7
         4 0 9
          4 9
           3
NUMERI MAGICI =  3 - 49
```

Una volta mandato in esecuzione il programma (niente bacchette magiche: bisogna prima copiarlo), questo propone un menu con due scelte: *chiave alfabetica* e *piramide magica*.

Entrambe le opzioni prevedono che venga immesso da tastiera un nome (proprio o comune, sono affari vostri), o una frase, purchè senza spazi e con soli caratteri alfabetici (p. nes. *carlo, banana, machebelcastello, dirondirondirondello*).

Unicamente per motivi estetici, se si è scelta la piramide magica, la lunghezza di quanto digitato non deve superare i 19 caratteri; in caso contrario, sarete comunque allertati.

Come si capirà meglio tra breve, questa limitazione è imposta dalla visualizzazione della "piramide" su uno schermo di 40 colonne.

Se non dovesse interessarvi l'aspetto estetico, o se (per il C/128) disponete di un monitor ad 80 colonne, basterà modificare il valore (19) presente in riga 270, o addirittura eliminare la linea stessa.

Dopo l'*Input*, penserà a tutto il programma.

Se avrete scelto la *Chiave Alfabetica*, dopo aver svolto i suoi calcoli, il computer si limiterà a stampare sul video uno o due numeri... magici, pronti per essere comunicati (d'urgenza) ad una qualsiasi ricevitoria.

Con l'*opzione 1*, viene invece elaborata sullo schermo un'immagine come quella che appare in figura, un triangolo (più che piramide) con vertice in basso, le cui due ultime righe rappresentano il risultato finale: i nostri agognati numeri.

Se provate a fornire lo stesso input per le due opzioni, potrete constatare come l'esito non sia quasi mai lo stesso.

Questo perchè i due metodi di "interpretazione" sono diversi; sta a voi trova-

re il più adatto (quello che vi sta più simpatico), o modificarne il funzionamento.

MAGICI ALGORITMI

Non crediate però, a dispetto della poca serietà dell'argomento, che i numeri forniti siano frutto della casualità.

Qualche *Randomize*, in questo caso, sarebbe stato più che sufficiente.

Entrambi i metodi, in pratica, poggiano su una stessa base, che poi elaborano diversamente. La base è costituita da un codice numerico, assegnato ad ogni lettera dell'alfabeto.

Nel listato, questo codice è inserito sotto forma di *Data* nell'ultima riga, e può essere eventualmente modificato.

Si tenga presente, in questo caso, che i numeri sono disposti ordinatamente in modo da sostituire, dalla *A* alla *Z*, tutto l'alfabeto (come da *Rem* soprastante).

Quindi, p. es., alla A corrisponde 2, alla B un 5, e così via.

Un'attenta consultazione di arcani e polverosi testi, ha messo in evidenza che di codici ce ne sono proprio tanti, ultimo tra tutti... quello Ascii (ne avete mai sentito parlare?).

La scelta dei codici adottati dal programma è tratta da un volumetto degli anni '50 (prezzo di copertina L. 60: troppo caro!), di autore non meglio identificato, ma zeppo di calcoli effettuati (addirittura) da *Rutilio Benincasa* (parente di Carneade?).

Tornando in tema, la *chiave alfabetica* funziona così (anche con un bagaglio di cognizioni minimo, le operazioni possono essere seguite sul listato):

Dopo aver sostituito ad ogni lettera il relativo numero, si esegue la somma di tutti gli ex-caratteri.

Tale somma va poi moltiplicata per 90, ed il risultato diviso per il numero delle lettere che compongono il nome o la frase.

L'esito di quest'ultimo calcolo, scremato da eventuali valori superiori a 90, contiene al suo interno i numeri da giocare (uno, o più di uno, a seconda dell'entità del numero).

La *Piramide Magica*, procede in maniera più divertente.

Su una riga (si veda figura), si scrivono tutti i numeri corrispondenti alle lettere, quindi si sommano due a due tra di loro.

Per esempio, in figura, si effettueranno le somme *6 + 8, 8 +8, 8 + 6,* ecc.

Il risultato, dal quale devono essere eliminate le eventuali decine (p. es. *6 + 8 =14*; risultato=*4*), va posto nella riga inferiore, tra i due addendi della somma.

Si procede così, riga dopo riga, fino a che non si ottiene un unico numero, che avrà tutti i crismi della... santità cabalistica, assieme a quello contenuto nella penultima riga, immediatamente sopra il vertice.

Come ovvio, vanno scartati eventuali risultati finali uguali a *0* o superiori a *90* (ci pensa il programma). I soliti smanettoni potranno ora spulciare il listato alla ricerca di modifiche e miglioramenti, superando le poche difficoltà legate, più che altro, all'esigenza di renderlo il più breve e compatto possibile. Non si dimentichi, però, la "delicatezza" dell'argomento.Stravolgimenti di calcoli millenari, potrebbero attirare su indifese tastiere le ire di *Nostradamus* & company!

```
100 REM -----------------------------------------------------
110 REM      C  O  M  M  O  D  O  R  C  A  B  A  L  A
120 REM          ------------------------------------
130 REM              PER C/64 E C/128 (MODO 128)
140 REM -----------------------------------------------------
150 :
160 DIMCM(26):FORY=1TO26:READCM(Y):NEXT
170 REM -------------  MENU DI SCELTA  ------------------
180 PRINTCHR$(147):PRINTSPC(7)"SCEGLI IL TIPO DI RESPONSO"
190 PRINT:PRINT,"1) PIRAMIDE MAGICA":PRINT
200 PRINT,"2) CHIAVE ALFABETICA":PRINT:PRINT,"3) FINE"
210 GOSUB550:IFA$<"1"ORA$>"3"THEN210
220 A=VAL(A$):IFA=3THEN580
230 REM -----------  INPUT E ASSEGNAZIONE CODICE  ---------
240 PRINTCHR$(147):PRINTSPC(3)"NOME O FRASE DA ANALIZZARE"
250 IFA=1THENPRINT"(MAX 19 CARATTERI E SENZA SPAZI) "
260 PRINT:INPUTNM$:PRINT:L=LEN(NM$):IFL<1THEN240
270 IFA=1ANDL>19THENPRINT"TROPPO LUNGO!":GOSUB540:GOTO240
280 DIMV(L):FORX=1TOL:Y=ASC(MID$(NM$,X,1))
290 IFY<65ORY>90THENPRINT"NON VALIDO!":GOSUB540:RUN
300 Y=Y-64:V(X)=CM(Y):IFA=2THEN390
310 REM ------------------   PIRAMIDE   -------------------
320 GOSUB510:PRINTV(X);CHR$(157);:NEXT
330 S=S+1:PRINT:PRINTSPC(S);:L=L-1
340 IFL=1THENN2$=STR$(V(1))+STR$(V(2)):N2=VAL(N2$)
350 IFL=0THENN1=V(1):GOTO450
360 FORX=1TOL:V(X)=V(X)+V(X+1):GOSUB510
370 PRINTV(X);CHR$(157);:NEXT:GOTO330 :REM STAMPA PIRAMIDE
380 REM --------------  CHIAVE ALFABETICA  ---------------
390 T=T+V(X):NEXT:T2=T*90:NR=INT(T2/L):IFNR=T2/LTHEN410
400 N3=T2-NR*L
410 NR$=STR$(NR):L=LEN(NR$):N2$=RIGHT$(NR$,2):N2=VAL(N2$)
420 IFL=4THENN1=VAL(LEFT$(NR$,2)):GOTO450
430 N1=VAL(LEFT$(NR$,3))
440 REM ---------------  NUMERI RISULTANTI  ---------------
450 IFN1=0THENN1=10
460 PRINT:PRINT"NUMERO/I MAGICI = "N1;
470 IFN2=<90ANDN2<>N1THENPRINT"-"N2;
480 IFN3<>0ANDN3<>N1ANDN3<>N2THENPRINT"-"N3
490 GOSUB540:RUN
500 REM ----------------  SUBROUTINES  -------------------
510 IFV(X)=>10THENV(X)=V(X)-10:GOTO510:REM-> TOGLIE DECINE
520 RETURN
530 REM --------------  ROUTINE DI ATTESA  ---------------
540 PRINT:PRINT:PRINT"PREMI UN TASTO"
550 GETA$:IFA$=""THEN550
560 RETURN
570 REM ----------------  FINE PROGRAMMA  -----------------
580 PRINTCHR$(147)"BUONA FORTUNA!!!!":END
590 REM ----------------- CODICE 'MAGICO' -----------------
600 :
610 REM A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
620 DATA2,5,6,1,3,7,5,2,4,7,7,0,8,3,8,6,9,4,1,9,0,5,0,3,9,1
630 END
READY.
```

MI SCUSI, MA LEI E' ASSICURATO ?

Ecco un semplice programma che mostra come fanno le varie Compagnie di Assicurazione a stabilire il "premio" annuale che tutti gli automobilisti sono tenuti a pagare

di Alessandro Marrazzo

Quando si parla dell'assicurazione per l'autovettura si pensa sempre a qualcosa di oscuro e incomprensibile da parte dell' utente, che si trova nella condizione di dover pagare il premio dovuto, senza sapere in base a quali regole è stato calcolato.

Tutto questo non è affatto vero perchè, come vedremo, il premio di una polizza auto viene determinato da regole molto semplici e, soprattutto, comprensibili da chiunque.

Il premio di una polizza di assicurazione è sempre *annuale* ed *anticipato*, anche se noi possiamo chiedere che venga frazionato in più rate.

I frazionamenti previsti sono tre: *semestrale, quadrimestrale* e *trimestrale*; in ognuno di questi casi al premio calcolato va però aggiunta una percentuale che, allo stato attuale delle cose, è stabilita nella misura del 3%, 4% e 5%, rispettivamente al frazionamento del premio in *2, 3* oppure *4* rate.

C'è poi da considerare il caso particolare di una *Polizza Temporanea*: si tratta cioè di una assicurazione con durata inferiore all'anno, con il premio che viene calcolato in base ai giorni di durata della copertura assicurativa, al quale va sommato un sovrappremio che, attualmente, viene calcolato nella percentuale del 15% rispetto al premio annuo.

Al momento della stipula del contratto di assicurazione, però, pur avendo bisogno di un certo tipo di frazionamento, potremmo avere anche l'esigenza che lo stesso abbia una determinata scadenza.

Ricorriamo un esempio per essere più chiari:

Assicuro la mia macchina a gennaio e richiedo un frazionamento semestrale (2); per mia comodità vorrei, però, che le due rate che ogni anno devo pagare non abbiano scadenza a ***gennaio*** *e a* ***luglio****, ma diciamo a* ***marzo*** *e* ***settembre****.*

Per ovviare a problemi di questo genere c'è la possibilità di pagare all'inizio un primo rateo (che, nel caso dell'esempio appena citato, sarebbe di tre mesi) e, alla scadenza dello stesso, cominciare il normale pagamento delle rate frazionate nel modo stabilito.

C'è poi il caso di una variazione in corso di contratto.

Il premio di *Responsabiltà Civile Auto* viene stabilito in base al numero di *cavalli fiscali* dell'autovettura; è chiaro, quindi, che se c'è una sostituzione del veicolo assicurato che comporta un passaggio di settore - e quindi una variazione del premio - bisogna eseguire dei calcoli, comunque molto semplici.

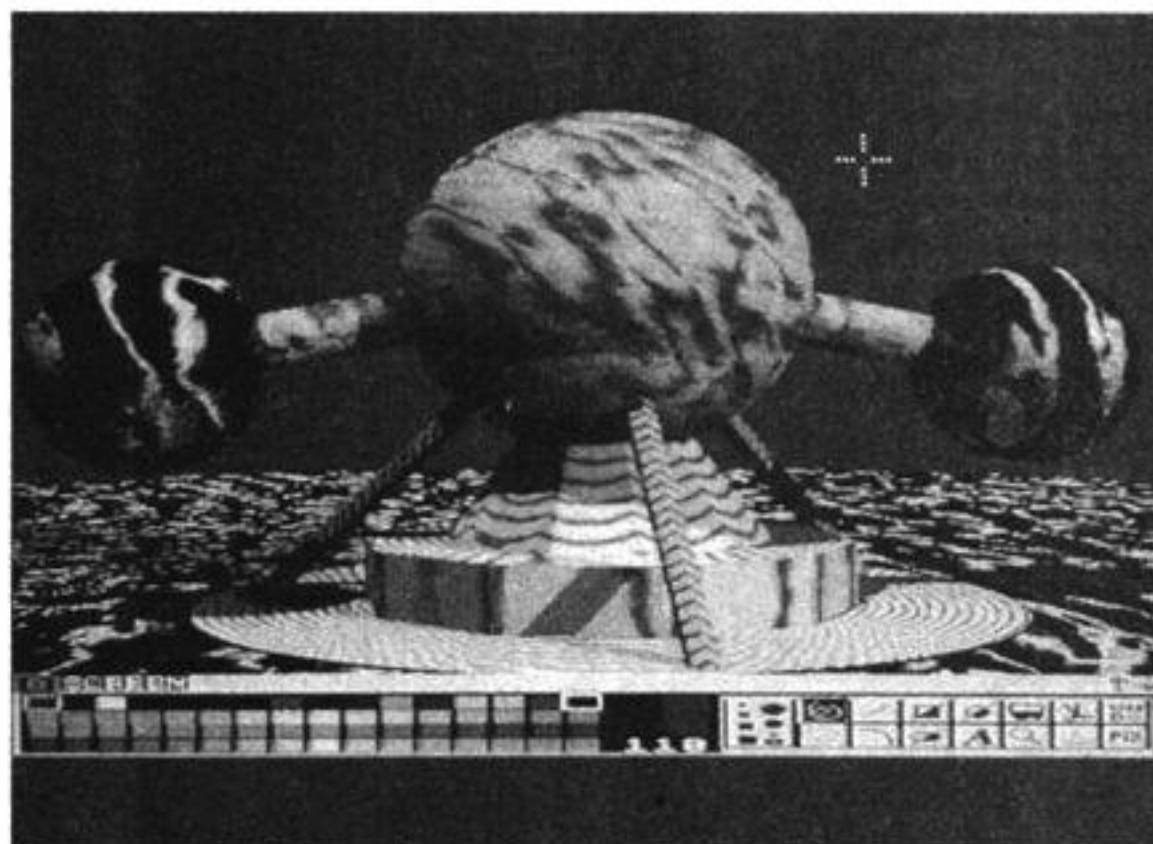
Bisogna cioè considerare la differenza tra il nuovo ed il vecchio premio per i giorni già pagati fino alla scadenza della prossima rata.

Nel caso di valore positivo il contraente della polizza di assicurazione dovrà provvedere ad una integrazione del premio, mentre, in caso di risultato negativo (abbiamo quindi cambiato la vecchia auto con una di minore potenza che, appartenendo ad un settore inferiore, paga di meno), sarà la Compagnia di Assicurazioni che rimborserà il premio pagato in più fino alla prossima rata; da quella scadenza in poi, si continuerà regolarmente a pagare le rate calcolate in base al settore di appartenenza della nuova autovettura.

TARIFFE

Per assicurare la nostra autovettura abbiamo a disposizione due principali tipi di tariffa: la tariffa *Fissa* e la tariffa *Bonus / Malus*; vediamo la differenza.

Nel primo caso, peraltro usato molto meno frequentemente del successivo, il premio annuo per la *responsabilità civile auto* risulta inizialmente inferiore rispetto all'altra tariffa, con la differenza, però, che non si gode di nessun beneficio durante il corso degli anni e che soprattutto, in caso di sinistro, bisogna provve-

dere di tasca propria al pagamento della *franchigia* stabilita in polizza.

Nel caso della formula tariffaria Bonus / Malus sono invece previste riduzioni o maggiorazioni del premio - rispettivamente in assenza o in presenza di sinistri durante i "periodi di osservazione" - articolate in *undici classi* di appartenenza, corrispondenti, ognuna, a diversi livelli di premio determinati secondo le tabelle di merito riportate in queste pagine.

All'atto della stipulazione il contratto, salvo che sia relativo ad un veicolo che ne sostituisca un altro assicurato nella forma Bonus / Malus (in questo caso si conserva la classe di merito in vigore), è assegnato alla *classe di merito 6* se relativo a un veicolo assicurato in precedenza con forma diversa, oppure alla *classe di merito 7* se relativo ad un veicolo immatricolato per la prima volta al P.R.A. (*Pubblico Registro Automobilistico*), o ad un veicolo assicurato per la prima volta dopo una *voltura* al P.R.A.

Ricordate, infine, chè se non siete soddisfatti della vostra attuale Compagnia di Assicurazioni avete la possibilità di cambiarla, disdettando il contratto in corso con lettera raccomandata, ***da spedire almeno tre mesi prima della scadenza*** dello stesso, e facendosi rilasciare l'*Attestazione Dello Stato Di Rischio* da consegnare al nuovo assicuratore, per conservare così le forme di beneficio e personalizzazione del premio accumulate fino a quel momento.

QUANTO COSTA

L'ammontare del premio di una polizza di assicurazione relativa ad una autovettura, varia, oltre che per le cause appena viste, anche per l'influenza di altri fattori, quali la ***provincia*** di immatricolazione del veicolo, il ***massimale*** (limite massimo indennizzabile dalla Compagnia di Assicurazioni in caso di sinistro) previsto, ecc...

Risulta evidente, quindi, che un programma idoneo a gestire dati per calcoli di questo genere, debba essere necessariamente qualcosa di estremamente professionale, sviluppato per essere utilizzato da macchine all'altezza del compito (cioè: Ms-Dos compatibili).

Il programma, che deve invece girare sul nostro piccolo C/64, deve quindi, per forza di cose, limitarsi al semplice calcolo del premio finale, frazionato e comprensivo di imposte (che ammontano al 12,5% del premio netto), determinato dai premi annui forniti dall'utente per ogni singola voce, data l'impossibilità di creare - con questo computer - archivi di dimensioni tali da contenere tutte le tabelle necessarie al programma stesso per stabilire i premi annui autonomamente.

Prima di poterlo utilizzare bisognerà riportare tutte le costanti che il programma utilizza, su un file sequenziale (su nastro o su disco) tramite il listato N. 2 (***Scrittura Dati***).

Questo per far sì che se un giorno dovessero aumentare o diminuire (?) le imposte stabilite dallo Stato per questo tipo di assicurazioni, o dovessero variare le percentuali per i vari frazionamenti previsti, non dovremo manomettere ogni volta il programma principale.

COME GIRA IL LISTATO

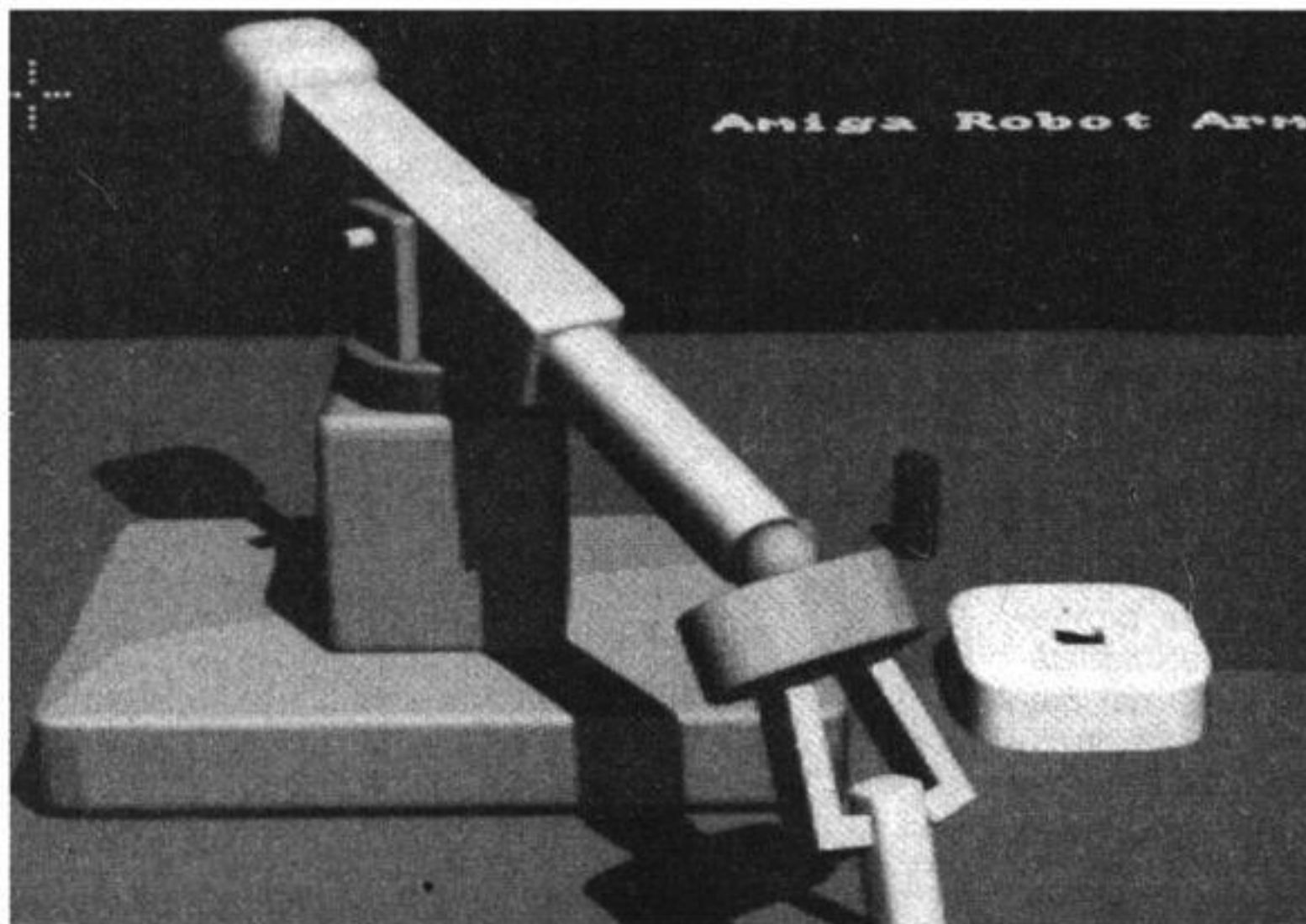
Basterà caricare (e far partire) il listato N. 2 rispondendo ai vari *Input* con i nuovi valori aggiornati. Le percentuali da inserire attualmente sono riportate in tabella.

Percentuale per:	**vari ab.**	**valo re**
Imposte	TA	12.5
Polizza temporanea	TE	8
Fraz. semplice	FS	3
Fraz. quadrimestrale	FQ	4
Fraz. trimestrale	FT	5

Dopo aver letto i dati ed aver calcolato ***Nf*** (coefficente dal quale, moltiplicandolo per il totale lordo, si ricavano le imposte) il programma chiede l'inserimento del premio di Responsabilità Civile annuo e al netto delle imposte; per rispondere a questa domanda - ed a quelle che seguiranno - abbiamo bisogno dell'aiuto del nostro agente di assicurazioni di fiducia, che potrà fornirci le tabelle dalle quali ricavare tutti i dati necessari.

Bisogna quindi inserire il valore da assicurare per la garanzia *incendio e furto* e, nel caso questo sia diverso da zero, il tasso per ogni mille lire assicurate applicato dalla nostra Compagnia.

Dobbiamo immettere poi per altre garanzie accessorie, se previste, il *premio annuo* (sempre al netto delle imposte) e infine il *numero delle rate* per le quali si intende frazionare il premio, oppure *5* se si vuole una Polizza Temporanea.

In questo caso, ignorando l'istruzione *On-Goto* di riga 2210 il programma continua il suo normale svolgimento: se abbiamo inserito in *V* (valore incendio e furto) un numero diverso da zero, dopo aver fatto apparire una segnalazione, si ritorna all'immissione dati, perchè di solito questa garanzia non viene concessa sulle polizze Temporanee.

Classe di merito	coeff. di determinazione del premio
1b	0.70
1a	0.70
1	0.70
2	0.75
3	0.80
4	0.85
5	0.92
6	1.00
7	1.15
8	1.32
9	1.52
10	1.75
11	2.00

In caso contrario si calcola il 15% del premio annuo di Responsabilità Civile (***R***) e si aggiunge alla somma calcolata dividendo il premio annuo per 360 (che per comodità si considera il numero dei giorni che conpongono un frazionamento annuale) e moltiplicandolo per i giorni di durata che abbiamo stabilito in riga 2250.

Stesso discorso anche per il premio Accessori (***A***) e si salta alla riga 2660.

Anche se avessimo scelto un altro frazionamento (1, 2, 3 oppure 4), dopo aver aggiunto ai premi calcolati le rispettive percentuali saremmo giunti alla riga 2660 dove sommiamo il premio Accessori (***A***) con il premio incendio e furto (***I***).

Si calcola quindi il totale netto, dal quale si ricavano le imposte (e quindi il totale lordo) e si salta alla riga 3300 dove otteniamo la visualizzazione di tutti i premi.

Arrivati a questo punto, abbiamo a disposizione varie opzioni, che andiamo ad esaminare una per una:

NERO-ALLINEA = 1 = Arrotondamento
(salto alla riga 2820)

Scegliendo questa opzione possiamo arrotondare a nostro piacimento il totale finale ottenuto e, dopo aver calcolato tramite ***Nf*** le nuove imposte (e quindi il totale netto), si ritorna alla riga 3300.

NERO-ALLINEA = 2 = Rateo
(salto alla riga 2930)

In riga 2990 poniamo a 1 un flag (***Sw***) che nel caso si trovasse in questo stato al momento della scelta di questa opzione, ci farebbe ritornare indietro da dove siamo venuti perchè significa che, per questo premio, abbiamo già calcolato un rateo (o una variazione).

In riga 3010 immettiamo la durata del rateo in giorni; nel caso questo risultassero in numero uguale o maggiore dei giorni che conpongono una rata del frazionamento scelto all'inizio, si tratterebbe di un'operazione non concessa; perciò, dopo aver fatto apparire un messaggio e aver riportato Sw a zero, torneremmo alla riga 3300.

Se invece è tutto regolare, si dividono i premi per 360 e si moltiplicano per i giorni di durata del rateo; i valori così ottenuti si moltiplicano per il frazionamento scelto all'inizio (abbiamo effettuato i calcoli con i premi già frazionati) e si passa alla riga 2730 per il calcolo dei totali e delle imposte.

NERO-ALLINEA = 3 = Variazione
(salto alla riga 3140)

In riga 3180 abbiamo il controllo del flag (già visto nel caso del rateo) che se risulta uguale a ***0*** viene posto a ***1*** in riga 3200.

A questo punto immettiamo i premi già corrisposti per il vecchio contratto - sia per la Responsabilità Civile che per gli Accessori - ed il numero dei giorni già pagati (quelli che mancano per arrivare allo scadere della prossima rata).

Calcoliamo la differenza tra i nuovi ed i vecchi premi, la dividiamo per 360, la moltiplichiamo per i giorni pagati; il risultato così ottenuto lo moltiplichiamo per il frazionamento scelto; andiamo di nuovo alla riga 2730 per calcolare i nuovi totali e le relative imposte.

NERO-ALLINEA = 4 = Altro Calcolo
(salto alla riga 1930)

Con questa opzione ci limitiamo a tornare alla riga 1930 per l'inserimento dei nuovi dati.

NERO-ALLINEA = 5 = Fine
(salto alla riga 3670)

Si salta alla riga 3670, da dove si arriva anche nel caso di errore da disco durante la lettura del file sequenziale (motivo della presenza delle righe 3710 - 3720), ed il programma termina.

Effettuando alcuni esempi pratici (e magari veri) poco alla volta vi renderete conto che, in effetti, si tratta di procedure molto semplici.

Certo che i programmi professionali, che potete vedere in funzione presso quasi tutte le agenzie di assicurazioni, sono un'altra cosa...

Classe di merito	Classe di collocazione per il periodo annuo successivo				
N. sinistri	0	1	2	3	4
1b	1b	1a	1	2	3
1a	1b	1	2	3	4
1	1a	2	3	4	5
2	1	3	4	5	6
3	2	4	5	6	7
4	3	5	6	7	8
5	4	6	7	8	9
6	5	7	8	9	10
7	6	8	9	10	11
8	7	9	10	11	11
9	7	10	11	11	11
10	8	11	11	11	11
11	9	11	11	11	11

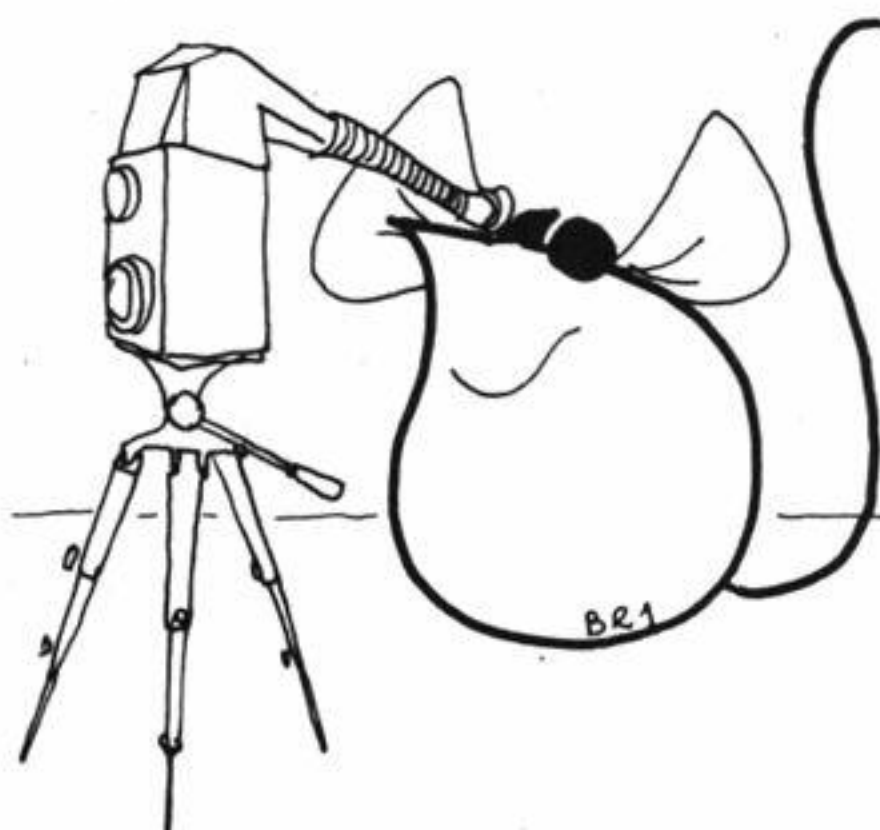
CONCLUSIONI

Lo scopo del presente articolo è solo quello di far capire in base a quali criteri viene determinato il costo dell'assicurazione per la nostra auto, e divulgare - soprattutto in vista di programmazioni future - il *trucco* del file sequenziale in cui scrivere i dati che potrebbero variare nel corso del tempo, senza per questo rendere inservibile il programma principale.

Procedura, quest'ultima, molto usata soprattutto in *package professionali* che, non potendo permettersi di diventare "vecchi", vengono all'occorrenza "ringiovaniti" tramite il semplice aggiornamento (o sostituzione) dei files necessari all'operazione.

```
1000 rem ************************
1010 rem *                      *
1020 rem *  premi polizze auto  *
1030 rem *                      *
1040 rem *      scritto da      *
1050 rem * alessandro  marrazzo *
1060 rem *                      *
1070 rem *       per c/64       *
1080 rem *                      *
1090 rem ************************
1100 :
1110 :
1120 :
1130 rem
1140 rem intro
1150 rem
1160 :
1170 :    print chr$(147)"leggo da nastro"
1180 :    input "o disco (n/d)";p$
1190 :    if p$<>"n" and p$<>"d" then 1170
1200 :    print
1210 :    input "nome file ";nf$
1220 :    if p$="d" then 1360
1230 :
1240 rem
1250 rem periferica - nastro
1260 rem
1270 :
1280 :    print
1290 :    print"inserisci il nastro"
1300 :    print"con il file ";nf$
1310 :    print"e premi un tasto"
1320 :    get a$:if a$="" then 1320
1330 :    open 2,1,0,nf$
1340 :    goto 1490:rem lettura dati
1350 :
1360 rem
1370 rem periferica - disco
1380 rem
1390 :
1400 : open 15,8,15:rem canale di errore
1410 :    print
1420 :    print"inserisci il disco"
1430 :    print"con il file ";nf$
1440 :    print"e premi un tasto"
1450 :    get a$:if a$="" then 1450
1460 :    open 2,8,2,"0:"+nf$+",s,r"
1470 :    gosub 3590:rem lettura errori
1480 :
1490 rem
1500 rem lettura dati
1510 rem
1520 :
1530 :    input# 2,ta
1540 :    if p$="d" then gosub 3590
1550 :
1560 :    input# 2,te
1570 :    if p$="d" then gosub 3590
1580 :
1590 :    input# 2,fs
1600 :    if p$="d" then gosub 3590
1610 :
1620 :    input# 2,fq
1630 :    if p$="d" then gosub 3590
1640 :
1650 :    input# 2,ft
1660 :    if p$="d" then gosub 3590
1670 :
1680 :    close 2
1690 :    if p$="d" then close 15
1700 :
1705 :    rem ricava imposte dal tot.fin.
1710 :    nf=ta*100/(ta*100+100)
1720 :
1730 rem
1740 rem copertina
1750 rem
1760 :
1770 :    poke 53280,0:rem cornice
1780 :    poke 53281,0:rem sfondo
1790 :    poke 646,12:rem caratteri
1800 :    print chr$(147)
1810 :    print chr$(14):rem minuscolo
1820 :
1830 print chr$(18);spc(13);"             "
1840 print chr$(18);spc(13);"POLIZZE AUTO"
1850 print chr$(18);spc(13);"             "
1860 print spc(253);"PROGRAMMA  PER"
1870 print spc(50);"IL CALCOLO DEI PREMI"
1880 print spc(255);"SCRITTO DA"
1890 print spc(50);"ALESSANDRO  MARRAZZO"
1900 :
1910 :    gosub 3510:rem ritardo
1920 :
1930 rem
1940 rem immissione
1950 rem
1960 :
1970 :    print chr$(147)
1980 :    r=0
1990 input "premio r.c. annuo netto £. ";r
2000 :    v=0
2010 input "valore incendio e furto £. ";v
2020 :    if v=0 then 2060
2030 :    t=0
2040 input "tasso  incendio e furto %. ";t
2050 :    i=int(v/1000*t)
2060 :    a=0
2070 input "premio acc. annuo netto £. ";a
2080 :    f=0
2090 input "frazion. (5=temporanea)    ";f
2100 :    if f<1 or f>5 then 2080
2110 :    on f goto 2300,2360,2430,2500
```

```
2120 :
2130 rem
2140 rem polizza temporanea
2150 rem
2160 :
2170 :    if v=0 then 2240
2180 :
2190 :    print
2200 :print "inc.e fur.non vale su temp."
2210 :    gosub 3510:rem ritardo
2220 :    goto 1930
2230 :
2240 :    print
2250 :    input "temp. con n. giorni ";g
2255 rem aggiunta 15 % del premio annuo
2260 :    r=int((r*te)+(r/360*g))
2270 :    if a then a=(a*te)+(a/360*g)
2280 :    goto 2660
2290 :
2300 rem
2310 rem frazionamento annuale
2320 rem
2330 :
2340 :    goto 2660:rem nessuna % aggiunta
2350 :
2360 rem
2370 rem frazionamento semestrale
2380 rem
2390 :
2400 : pf=fs:rem aggiunta sui premi del 3 %
2410 :    goto 2570
2420 :
2430 rem
2440 rem frazionamento quadrimestrale
```

```
2450 rem
2460 :
2470 : pf=fq:rem aggiunta sui premi del 4 %
2480 :    goto 2570
2490 :
2500 rem
2510 rem frazionamento trimestrale
2520 rem
2530 :
2540 :    pf=ft:rem aggiunta sui premi del 5 %
2550 :    goto 2570
2560 :
2570 rem
2580 rem sovrappremio per frazionamento
2590 rem
2600 :
2610 :    r=int((r+r*pf)/f):rem resp. civile
2620 :    i=int((i+i*pf)/f):rem inc.+fur.
2630 :    a=int((a+a*pf)/f):rem accessori
2640 :    rem goto 1750
2650 :
2660 rem
2670 rem somma premi rischi accessori
2680 rem
2690 :
2700 :    rc=r:rem premio resp. civile
2710 :    ac=a+i:rem premio rischi access.
2720 :
2730 rem
2740 rem calcolo totali e imposte
2750 rem
2760 :
2770 :    t1=rc+ac:rem totale netto
2780 :    im=int(t1*ta):rem imposte
2790 :    t2=t1+im:rem totale finale
2800 :    goto3300:rem visualizzaz. premi
2810 :
2820 rem
2830 rem arrotondamento
2840 rem
2850 :
2860 :    print
2870 :    input "nuovo totale finale £. ";t2
2880 :    im=int(t2*nf)
2890 :    t1=t2-im
2900 :    rc=t1-ac
2910 :    goto 3300:rem visualizzazione premi
2920 :
2930 rem
2940 rem rateo
2950 rem
2960 :
2965 rem rateo o variazione gia' calcolati
2970 :    if sw=1 then 3470
2980 :
2990 :    sw=1
3000 :    print
3010 :    input "rateo di giorni ";g
3020 :    if g<360/f then 3100
3030 :
3040 :    print
3050 :    print "rateo minore del fraz."
3060 :    gosub 3510:rem ritardo
3070 :    sw=0
3080 :    goto 3300
3090 :
3100 :    rc=int(rc/360*g)*f
3110 :    if ac then ac=int(ac/360*g)*f
3120 :    goto 2730:rem calcolo totali
3130 :
3140 rem
3150 rem variazione
3160 rem
3170 :
3175 rem rateo o variazione gia' calcolati
3180 :    if sw=1 then 3470
3190 :
3200 :    sw=1
3210 :    print
3220 : input "vecchio premio r.c. £. ";vr
3230 : input "vecchio premio acc. £. ";va
3240 : input "giorni gia' pagati      ";gp
3250 :
3260 :    rc=int((rc-vr)/360*gp)*f
3270 :    ac=int((ac-va)/360*gp)*f
3280 :    goto 2730:rem calcolo totali
3290 :
3300 rem
3310 rem visualizzazione premi
3320 rem
3330 :
3340 :    print chr$(147);
3350 :    print "premio resp. civile  £.";rc
3360 :    print "premio rischi acces. £.";ac
3370 :    print "totale netto         £.";t1
3380 :    print "imposte              £.";im
3390 :    print "totale finale        £.";t2
3400 :    print
3410 :    print "1 - arrotondamento"
3420 :    print "2 - rateo"
3430 :    print "3 - variazione"
3440 :    print "4 - altro calcolo"
3450 :    print "5 - fine"
3460 :    print
3470 :    input "opzione scelta ";o
3480 :    if o<1 or o>5 then 3470
3490 :    on o goto 2820,2930,3140,1930,3670
3500 :
3510 rem
3520 rem ritardo
3530 rem
3540 :
3550 :    for x=1 to 3000
3560 :    next
3570 :    return
3580 :
3590 rem
3600 rem lettura errori su disco
3610 rem
3620 :
3630 :    input# 15,en,em$,et,es
3640 :    if en=0 then return
3650 :    print en;em$;et;es
3660 :
3670 rem
3680 rem fine programma
3690 rem
3700 :
3710 :    close 2
3720 :    if p$="d" then close 15
3730 :    end
```

```
1000 rem ***********************
1010 rem *                     *
1020 rem *    scrittura dati   *
1030 rem *                     *
1040 rem *                     *
1050 rem * alessandro marrazzo *
1060 rem *                     *
1070 rem *      per c/64       *
1080 rem *                     *
1090 rem ***********************
1100 :
1110 :
1120 :
1130 rem
1140 rem intro
1150 rem
1160 :
1170 :    print chr$(147)"Scrittura dati"
1180 :    input "su nastro o su disco ";p$
1190 :    if p$<>"n" and p$<>"d" then 1170
1200 :    print
1210 :    input "nome file ";nf$
1220 :    if len(nf$)>16 then 1210
1230 :    if p$="d" then 1350
1240 :
1250 rem
1260 rem periferica - nastro
1270 rem
1280 :
1290 :    print
1300 :    print"inserisci un nastro"
1310 :    print"e premi un tasto"
1320 :    geta$:if a$="" then 1320
1330 :    goto 1470:rem input dati
1340 :
1350 rem
1360 rem periferica - disco
1370 rem
1380 :
1390 :    open 15,8,15:remcanale di errore
1400 :    print
1410 :    print"inserisci un disco"
1420 :    print"e premi un tasto"
1430 :    geta$:if a$="" then 1430
1440 :    open 2,8,2,"@0:"+nf$+",s,w"
1450 :    gosub 2050:rem lettura errori
1460 :
1470 rem
1480 rem input dati
1490 rem
1500 :
1510 :    print chr$(147)
1520 :    print "percentuale per"
1530 :    input "le imposte ";ta
1540 :    ta=ta/100
1550 :
1560 :    print
1570 :    print "percentuale per"
1580 :    input "polizza temporanea ";te
1590 :    te=te/100
1600 :
1610 :    print
1620 :    print "percentuale per"
1630 :    input "frazion. semestrale ";fs
1640 :    fs=fs/100
1650 :
1660 :    print
1670 :    print "percentuale per"
1680 :    input "fraz. quadrim. ";fq
1690 :    fq=fq/100
1700 :
1710 :    print
1720 :    print "percentuale per"
1730 :    input "fraz. trimest.";ft
1740 :    ft=ft/100
1750 :
1760 rem
1770 rem scrittura dati
1780 rem
1790 :
1800 :    if p$="n" then open 2,1,1,nf$
1810 :
1820 :    print# 2,ta
1830 :    if p$="d" then gosub 2050
1840 :
1850 :    print# 2,te
1860 :    if p$="d" then gosub 2050
1870 :
1880 :    print# 2,fs
1890 :    if p$="d" then gosub 2050
1900 :
1910 :    print# 2,fq
1920 :    if p$="d" then gosub 2050
1930 :
1940 :    print# 2,ft
1950 :    if p$="d" then gosub 2050
1960 :
1970 rem
1980 rem fine
1990 rem
2000 :
2010 :    close 2
2020 :    if p$="d" then close 15
2030 :    end
2040 :
2050 rem
2060 rem lettura errori su disco
2070 rem
2080 :
2090 :    input# 15,en,em$,et,es
2100 :    if en=0 then return
2110 :    print en,em$,et,es
2120 :    goto 1970
```

Commodore

COMPUTER CLUB

La rivista degli utenti di sistemi Commodore

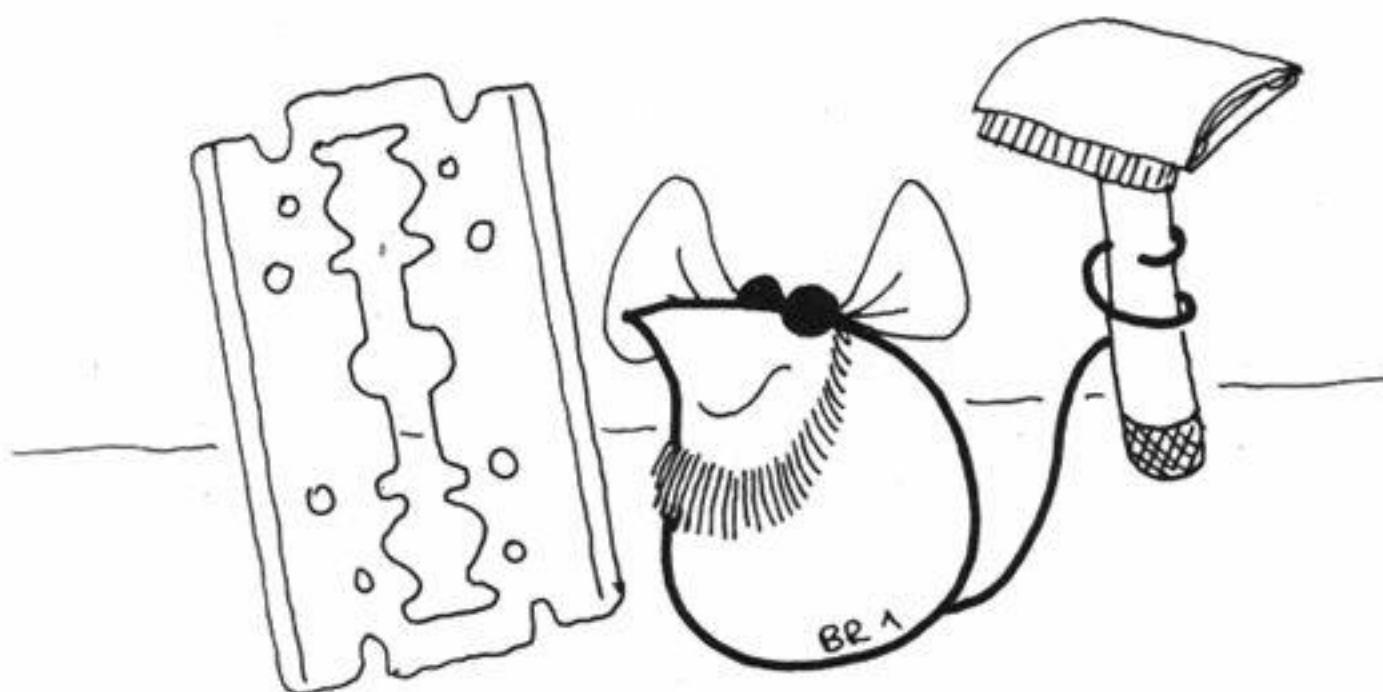
STARBYTE

FAST LANE!

The JETSONS

AMIGA

Commodore MODEL 1081

POWER

BATTLE VALLEY

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade
Softhouse: Hewson

Il governo USA ha una missione speciale per noi mercenari: farci sterminare un gruppo di pericolosi terroristi.

Alcuni terroristi hanno rubato due missili balistici di medio raggio, formalmente vietati dal trattato di pace stipulato dal governo. I loschi figuri chiedono la scarcerazione di tutti i loro compagni detenuti dietro la minaccia di usare i missili.

Il gioco

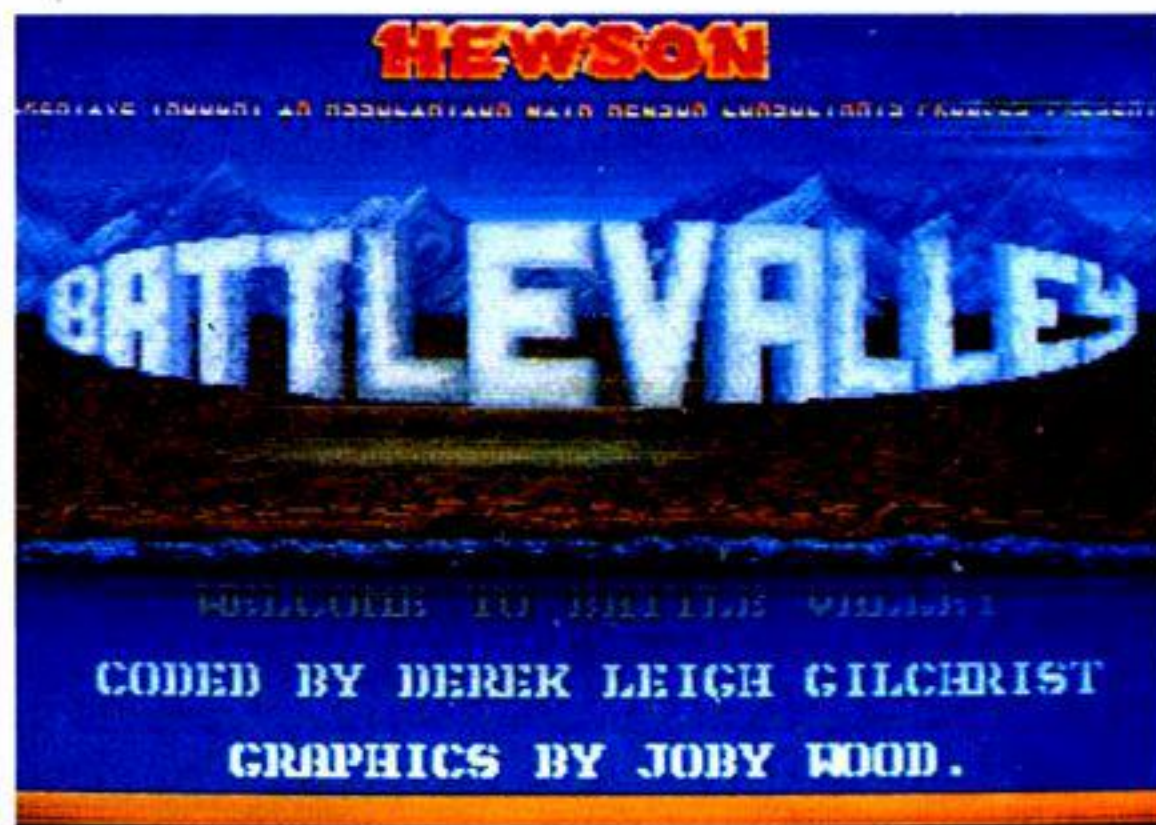
Non vi è molto da dire a chi già conosce la versione originale, per il C/64, estremamente avvincente e con grafica di buon livello, soprattutto considerando le prestazioni della macchina ad otto bit. La versione Amiga conserva tutti gli aspetti positivi, rapportati alle prestazioni consentite dal computer.

La missione inizia con semplicità, consentendo di penetrare progressivamente nelle difese avversarie, con ostacoli che a mano a mano diventano sempre più difficili da superare. Si parte dal quartier generale con un elicottero od un carrarmato, dirigendosi verso sinistra o verso destra. Lo scopo è di fare saltare le armi di difesa dei terroristi e di recuperare pezzi per costruire un ponte che consenta al nostro carrarmato di avanzare. Solo quest'ultimo tipo di arma, infatti, può fare saltare in aria la base dei terroristi, perciò è salutare usare nelle prime fasi l'elicottero per aprirgli la strada.

La Tecnica

La musica di titolo non è granchè, mentre gli effetti sonori sono molto buoni, in particolare le esplosioni, sebbene siano troppo semplificati. Lo scrolling, punto cardine del gioco, è di buona fattura; le scelte dei colori, considerato che il paesaggio è generato algoritmicamente (e non mappato in memoria), è certamente ben realizzato.

Il Voto

Idea non originale, scarsi incentivi, discreta giocabilità. 6 e mezzo.

PREDATOR

Arnie Schwarzenegger non è proprio riportato in tutto il suo splendore muscolare nell'indaffarato sprite di questo videogioco, ispirato all'omonimo film di fantascienza violenta.
Per chi non lo avesse visto, ricordiamo che mister *Olympia* deve qui recuperare un elicottero ed alcuni uomini rapiti da un alieno.
Il gioco si snoda in quattro fasi, in pratica sempre in una foresta, con il protagonista che cambia armi con la stessa velocità con cui cambierebbe i fazzoletti nel taschino, procedendo sempre di corsa da sinistra verso destra sullo schermo con scrolling continuo e sparando a tutto ciò che si muove.

Le insidie sono di parecchi generi, dagli artiglieri agli animali feroci della jungla. Le armi che si trovano, procedendo nelle varie fasi,conferiscono il tipo di potenza od elasticità di fuoco necessarie per affrontare con successo gli ostacoli susseguenti: non raccattandone qualcuna ci si trova rapidamente fuori fase, anche perchè è l'unico modo per non terminare le munizioni, non essendo previsti approvvigionamenti volanti di soli proiettili.

Dal film ai nostri schermi; ma era meglio in pellicola...

Computer: Amiga inespanso
Gestione: Joystick
Tipo: arcade bellico
Softhouse: Activision

Tecnica

La grafica di sfondo potrebbe andare bene, ma quella animata è ai limiti del disgustoso. Anche tutti i particolari aggiunti, come ad esempio i cadaveri e le armi dell'alieno, sono di qualità molto,

ma molto peggiore di quanto eravamo abituati a vedere sul C/64. I suoni sono della stessa infima qualità. Lo scrolling orizzontale, che dovrebbe essere facilitato dalla semplicità della grafica e dal basso numero di particolari e colori della scenografia di fondo, è invece scattoso e completamente inaccettabile.

Voto

L'idea era semplice, ma la realizzazione è pessima. 4.

FAST LANE

L'ennesimo figlio di *Pole Position* e *Chequered Flag*, che si affianca senza particolare originalità, nè molte innovazioni, a tanti altri programmi analoghi, sebbene in corse di gruppo C.

Il Gioco

Una competizione automobilistica non si basa certamente soltanto sull'abilità del pilota nell'affrontare le curve, schiacciare l'acceleratore e manovrare il cambio, ma bensì sulla perfetta preparazione dell'auto dal punto di vista tecnico e tecnologico, l'organizzazione dei box, le istruzioni al pilota eccetera eccetera.

In questo programma si controlla una *Cosworth SE89C* modificata per corse agonistiche.

Lo scopo è di vincere il campionato mondiale di categoria.

Il punto di forza del programma è nella pretesa di ricalcare quanto più possibile le effettive procedure di corsa, non tanto la fedeltà e la velocità della grafica, che difatti non è certamente delle migliori in circolazione. Ciò significa che, anche volendo soltanto fare qualche giro di prova prima di cimentarsi nelle qualificazioni per il campionato mondiale, non si può partire con la macchina semplicemente premendo sull'acceleratore come in qualunque altro gioco del genere.

Bisogna infatti accendere manualmente il quadro elettrico, tirare lo starter, verificare i vari quadranti ed infine mettere in moto agendo sul motorino di avviamento.

Prima di salire in macchina bisogna comunque stabilire tutte le sue parti in dettaglio: tipo di spoiler, pneumatici, fari, carburazione e tanto altro ancora.

Questo va fatto di volta in volta, perchè le condizioni di corsa variano: si può correre una volta di notte, un'altra volta con la strada ghiacciata ed un'altra ancora su di un percorso con pochi rettilinei. Imparare come si prepara adeguatamente il veicolo,

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Corsa automobilistica
Softhouse: Artronic

Le corse automobilistiche sono uno degli sport più emulati su Amiga

in funzione del tipo di corsa che attende, è quindi importante quanto il guidare efficientemente.

La fedeltà di simulazione si riflette anche sulla guida vera e propria.

Qui, ad esempio, non è pensabile andare fuori strada ad alta velocità e rientrare come se nulla fosse accaduto.

Anzi, come guidando una vera macchina, spesso ci si troverà a spegnere il motore o a sentire le gomme urlare dal dolore quando si prende una curva a velocità troppo sostenuta.

Tecnica

La grafica è sufficientemente gradevole, sebbene i dettagli a bordo strada ed i particolari delle altre vetture non siano nulla di entusiasmante.

La velocità e la fluidità dello scrolling è buona, così come gli effetti sonori, che risultano di qualità superiore alla media di questo tipo di giochi.

La maggior parte di tecnica scientifica è stata probabilmente introdotta per simulare le reali prestazioni di una macchina da corsa, molto più di quanto normalmente previsto da giochi di questo tipo.

Voto

Buon realismo dal punto di vista tecnico, un po' meno graficamente.

Non molte innovazioni, ma comunque un programma longevo come interesse. 6/7.

FORGOTTEN WORLDS

Il classico gioco dell'uomo del futuro che, su una ciambella volante, deve distruggere tutto quello che incontra sparando con il bazooka che porta sotto il braccio.

Il gioco

Da soli od in coppia si deve affrontare una torma di alieni che hanno invaso il pianeta: mai che vengano solo a prendere un caffè da noi. Il gioco si svolge, a tratti, da sinistra verso destra e dall'alto verso il basso, quindi è necessario un buon colpo d'occhio per "commutare" istantaneamente da uno all'altro e non spararsi un colpo di bazooka sull'alluce.

Mentre si procede si incontra di tutto: draghi, caverne, laghi sotterranei con strane e pericolose piattaforme, come è buona regola in questo tipo di giochi. Si deve ricordare di non cozzare contro gli ostacoli, di sparare contro tutto ciò che si muove (od è pericoloso) e di raccogliere gli *Zenny blu* lasciati da alcuni nemici esplosi.

Questo fa guadagnare del denaro e, quando si può entrare in uno dei grandi magazzini che ogni tanto appare dal fondo dello schermo, si può spenderli per fare scorta di armi e munizioni.

La tecnica

Il gioco ha un buon scroll con sprites ben dimensionati, mentre l'animazione non è sempre ineccepibile. L'interfacciamento con il joystick non è dei migliori, così come gli effetti sonori, mediocri come tutto il resto.

Il voto

Una conversione mediocre di un buon gioco. 6 più.

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade multischermo
Softhouse: Virgin

L'ennesimo gioco spara e fuggi; attenti a

DRAKKHEN

Computer: Amiga inespanso
Gestione: Mouse e tastiera
Tipo: avventura animata
Softhouse: Infogrames

Storia di un altro mondo

In un'altra era di un altro mondo i *Drakkhen*, perfidi esseri squamosi, hanno raggiunto il potere da quando tutti i dragoni del pianeta sono stati uccisi.

Scopo del gioco è controllare quattro prodi avventurieri alla ricerca dei loro gioielli che potrebbero rimettere in vita il dragone e scacciarli per sempre.

Il gioco

Si può giocare in modo "singolo" od in modo "di gruppo". Nel primo caso si ha sullo schermo una visione separata dei quattro protagonisti per poterli manovrare efficacemente durante esplorazioni, raccolta di oggetti, combattimenti, eccetera. Nel modo "di gruppo" si ha una visione globale del gruppo, che consente di spostarli insieme.

I combattimenti sono assai frequenti, dal momento che vi sono circa 150 differenti tipi di avversari, e possono anch'essi essere effettuati da un solo elemento della compagnia oppure da tutti i personaggi insieme.

Il controllo è molto semplice, perchè lo schermo è suddiviso in varie finestre, delle quali una contiene i gadget di manovra principali (ma è comunque indispensabile usare anche la tastiera).

La tecnica

Gli effetti sonori sono terrificanti e di grande atmosfera. La grafica è molto buona ed estremamente curata nei dettagli. Per esempio, è notevole osservare la serie di movimenti compiuti da un personaggio che deve impugnare un'arma dopo averla sfilata dalla bisaccia posteriore.

Il voto

Originale, con molti incentivi e tanti personaggi diversi, tutti da scoprire. 7 e mezzo.

CLOWN 'O' MANIA

Ecco un videogame che destinato ad esser venduto in moltissime copie, come il leggendario *Manic Miner* di Matthew Smith.

Il gioco

Beppo è un simpatico pagliaccio di circo alle prese con una serie di piattaforme tridimensionali sulle quali sono collocati dei diamanti (che valgono dieci punti) e dei cristalli sferici che valgono un solo punto. Come prevedibile, Beppo deve raccoglierne il maggior numero possibile, evitando di precipitare dalle piattaforme e di scontrarsi con i vari mostriciattoli, sempre più cattivi e veloci, che incontra.

L'accesso alle varie piattaforme avviene tramite scalette e tunnel (per scendere) oppure con trampolini e vulcani (per salire). Sono previsti anche dei futuristici teletrasportatori per coprire distanze più ampie e, soprattutto, seminare i mostri che sono alle calcagna.

Ad un certo punto si nota anche che le caselle calpestate cambiano di colore. In questo caso si vince un bonus di diecimila punti se si termina il livello raccogliendo tutte le pietre preziose possibili e cambiando tutte le caselle ad uno stesso colore (cosa che rasenta i limiti del possibile, soprattutto quando si hanno dei mostri che vi inseguono).

Sono previsti anche oggetti di varia natura e funzioni: *maschere* da clown che danno vite bonus, *lame* di rasoio per uccidere i mostri, *molle* per saltare gli spazi vuoti e via fantasticando. Da notare che questi oggetti spesso si muovono sullo schermo, e velocemente anche!

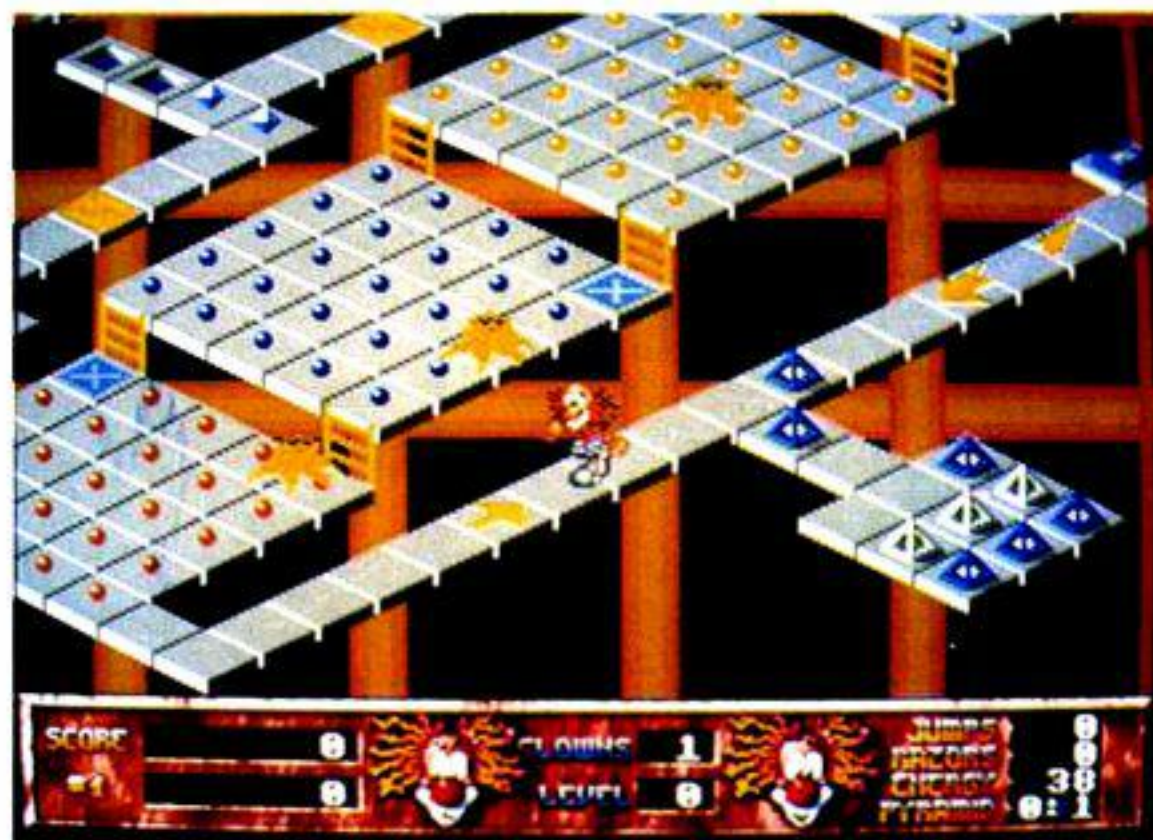
Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade a piattaforme
Softhouse: Starbyte

Un gioco di vecchio stile, che rievoca il sapore dei primi videogiochi

La tecnica

La grafica è semplice, ma coloratissima e divertente. Lo sprite di Beppo è ben animato ed imparare a controllarlo risulta difficile ai primi tempi (qui sta proprio uno degli incentivi del gioco), ma è comunque sempre pronto ai comandi anche quando lo schermo è affollatissimo. Lo scrolling è valido. I numerosi sprites animati forniscono incentivi ed interesse al gioco.

Voto

Semplice, ma ben realizzato. Richiede azione e riflessione. 7 1/2.

FUTURE WARS

La macchina del tempo è l'ingrediente di tanti giochi d'avventura, e Future Wars, dopo *Chrono Quest* e con *Time*, non è da meno.

Il gioco

Questa avventura grafica segue da vicino gli standard della *Sierra On Line*, autrice di titoli celeberrimi come *Leisure Suit Larry, Space Quest* e *Police Quest* per quanto riguarda la complessità delle animazioni ed il sottile senso di humor che pervade molte situazioni, ma assorbendo da prodotti *Cinemaware* una buona fetta di preziosità grafica per quanto riguarda gli sfondi. La trama dice che il pianeta Terra è stato sotto attacco per molti anni, ma grazie al sistema *SDI* ora è al riparo dagli alieni.

Gli scienziati hanno però scoperto che gli alieni, grazie ad una macchina del tempo, sono ritornati all'anno *1304*, non si sa per quale motivo. In effetti non si dice null'altro, a cominciare da come entriamo noi e come possiamo risolvere questo problema ciclopico. Tutto è da scoprire giocando, con molta pazienza e poco pensiero laterale, in effetti.

I puzzle richiedono poca riflessione, generalmente, e chi è pratico di avventure sarà già abbastanza smaliziato da risolvere senza indugi parecchie situazioni. Inoltre, dato che il numero di verbi di azione è limitato, si può in casi disperati procedere per tentativi, che comunque sono in numero finito a differenza dei normali adventure con input da tastiera.

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Avvantura grafica
Softhouse: Delphine

Facciamo un salto nel futuro per verificare se ci sarà davvero la pace...

La tecnica

Questo programma è distribuito in Italia dalla Lago/Soft Mail in edizione italiana, con tutte le frasi ed i manuali tradotti.

Tutti gli ordini vengono impartiti tramite cinque *gadget* a stringa, che nella versione italiana sono chiamati: *Esamina, Prendi, Inventario, Adopera, Usa* e *Parla*. Dopo avere clickato sul verbo o l'azione bisogna indicare sullo schermo l'oggetto interessato con un altro click del mouse.

Il voto

Un gioco di sicuro successo, grazie anche all'edizione italiana, pregevolmente rifinito, lungo da completare e comunque appassionante. 8 e mezzo.

AQUANAUT

Il gioco

Nella prima sezione del gioco (che è fornito su tre dischetti) il protagonista, *Ric Flair*, si tuffa nell'immensità dell'oceano in scrolling continuo. Come nell'antico videogioco *Scuba Dive*, deve nuotare sbattendo le pinne e fiocinando tutti i pesci che vede, evitando di sbatterci addosso pena la morte.

Troviamo meduse, squali, naselli, mine sottomarine ed altro ancora.

Si devono ricercare qui dei pezzi di attrezzatura indispensabile per proseguire nel gioco. La successiva sezione si svolge attraverso caverne sottomarine che portano ad *Atlantide*. Anche qui bisogna raccattare pezzi utili per il completamento dell'impresa. La terza ed ultima fase di svolge nei corridoi di Atlantide, infestati da alieni di varie specie. Qui lo scopo è di trovare i quattro Atlantidei sopravvissuti, raccogliendo ed usando appropriatamente gli oggetti disseminati, come al solito, per i vari schermi. In tutte e tre le sezioni ci si trova a dovere risolvere puzzle, talvolta proprio per raccogliere i pezzi necessari al completamento della missione. Si tratta di problemi piuttosto semplici, che in genere non fanno sprecare troppe vite nelle prime partite.

La tecnica

La sezione graficamente migliore è probabilmente la prima, dotata di scrolling continuo e di una quantità di esseri acquatici ben mossi. Le altre due fasi sono più "normali", anche se alcuni sprites riservano piacevoli sorprese dal punto di vista tecnico.

Gli effetti sonori non sono particolarmente entusiasmanti ed il controllo via joystick non sempre è precisissimo.

Il voto

Un gioco senza troppe pretese, ben realizzato e che può risultare interessante per un certo tempo, pur senza entusiasmare. 6 e mezzo.

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade
Softhouse: Addictive

Tanto per cambiare, la terra sta per essere invasa, ma questa volta i cattivi arrivano dalla città di Atlantide, sommersa dalle acque.

THE JETSONS

"*I Jetsons*" è il nome originale di quei personaggi di fumetti e di cartoni animati prodotti dalla *Hanna & Barbera*, da noi noti come *I Pronipoti*, della stessa famiglia dei *Flintstones* ovvero *Gli Antenati*.

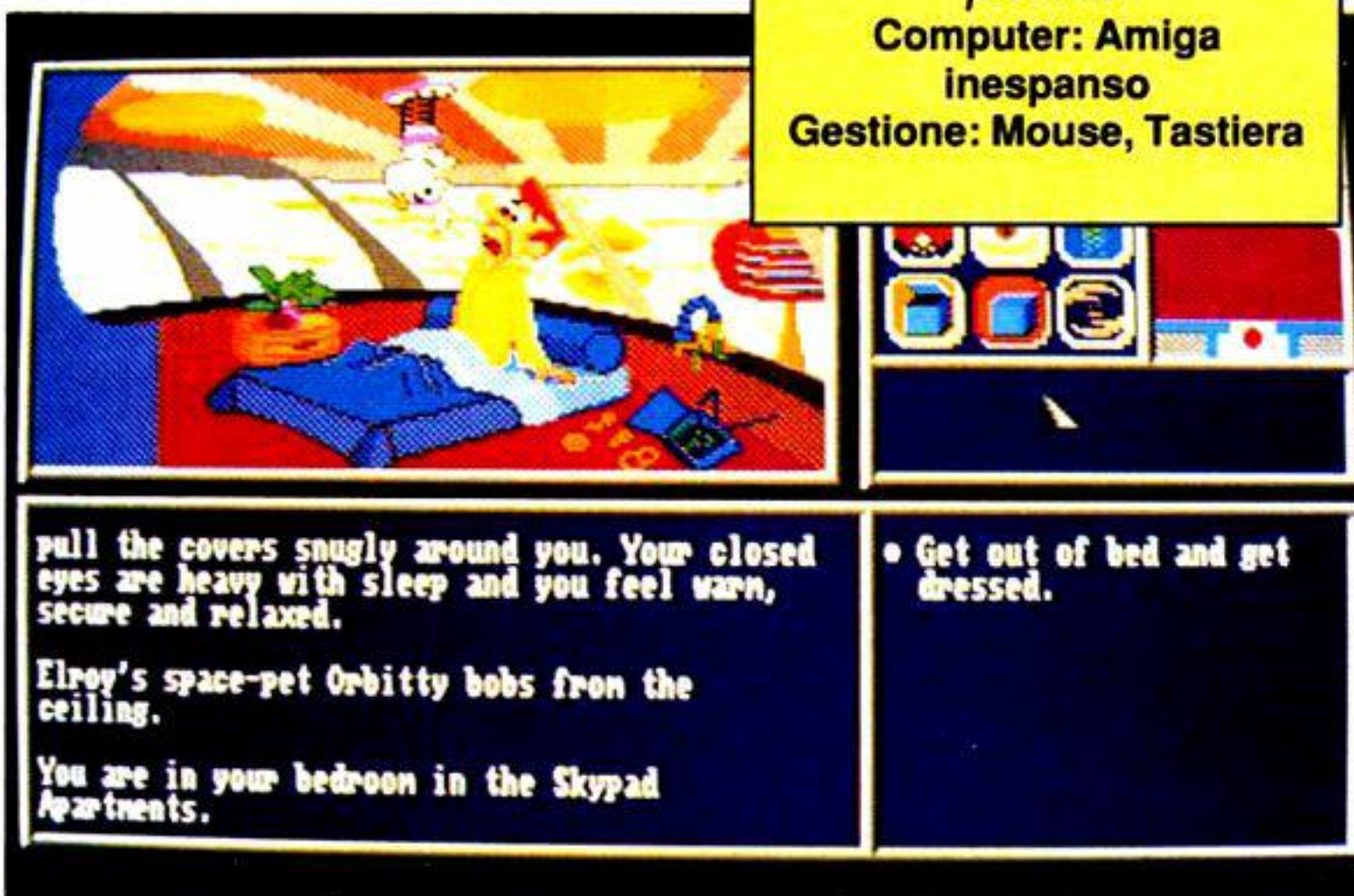
La famigliola vive nel futuro, con gli stessi canoni e molte delle stesse paranoie che abbiamo noi nel presente. Ciò fornisce la base per parecchie situazioni umoristiche e divertenti sia nel fumetto originale che nel videogioco ad esso ispirato.

Il capofamiglia è *George*, un simpatico ingenuo che lavora come pendolare, con la vivace e graziosa mogliettina *Jane* che stira e fa la casalinga con tante comodità futuristiche ed egualmente tanto stress. Altri componenti del nucleo famigliare sono il simpatico cagnolotto *Astro* ed il figlioletto *Elroy*, che sono specializzati nel combinare guai di portata biblica tra le mura di casa, specialmente quando sono insieme, oltre alla originale Jane.

Il personaggio che si controlla direttamente da programma è *George Jetsons* ed il suo scopo è di risolvere i piccoli problemi della vita quotidiana, e soprattutto conservare il posto di lavoro sopportando le angherie del suo datore, *Mr. Spacely*, e di competere con l'arci rivale di sempre, *Mr. Cogswell*.

Lo schermo offre quattro finestre: una presenta la grafica, eventualmente animata, al di sotto della quale troviamo i commenti (scritti, ovviamente, in inglese) su quanto sta accadendo; accanto c'è la serie di gadget per le azioni e sotto le azioni ed eventuali frasi che si possono dire o fare semplicemente clickando col mouse sopra di esse. Risulta così estremamente semplice ed immediato giocare senza dovere digitare alcunchè, nè scervellarsi a ricercare sul vocabolario inglese.

Un fumetto "trasportato" su Amiga con notevole perizia

Computer: Amiga inespanso
Gestione: Mouse, Tastiera

Tecnica

L'interfaccia grafica ad icone per giochi di avventura ha qui fissato un altro optimum qualitativo.

Mai avevamo visto un sistema così efficiente, comodo e graficamente prezioso allo stesso tempo.

La qualità dei fumetti di Hanna e Barbera è stato conservato appieno, inserendo talvolta delle semplici animazioni molto gradevoli e divertenti. Anche lo humor delle situazioni è stato conservato fedelmente.

Manca un sonoro di qualità, del resto poco utile in giochi di questo tipo; comunque la colonna sonora è ineccepibile.

Voto

Un gioco di avventura per tutti, semplice, divertente e con molti incentivi. 7 1/2.

SPACE ACE

Nessun videogiocatore che si rispetti può essersi dimenticato della rivoluzione apportata da un certo *Don Bluth* nei primi anni ottanta, quando produsse per la *Atari* il primo videogioco con base di dati su videodisco, creando così una specie di cartone animato interattivo. A tutt'oggi la qualità della grafica della produzione Don Bluth, *Dragon's Lair* per chi fosse smemorato, non è stata superata da nessun Coin Op, anzi recentemente è stato prodotto *Space Ace*, che lo ha bissato.

Ambedue i programmi sono stati realizzati anche per Amiga, come costosi programmi dotati di 4/6 dischetti, richiedendo almeno un megabyte di memoria *RAM* per potere funzionare.

Mentre Dragon's Lair era ambientato negli *Anni Scuri* e consisteva nel recupero di una bionda intrappolata in un castello da parte di un certo *Dirk Daring*, Space Ace è ambientato, all'opposto, nel futuro. L'eroe qui è *Ace*, che deve contrastare l'alieno tiranno *Borf* (che non è verde, come un marziano, ma blu), rapitore della sua bella *Kimberly* (queste donne procurano sempre guai nei videogiochi, e non soltanto in questi...).

Il gioco si svolge in 33 scene differenti, tutte basate sul semplice concetto di liberazione e lotta. Il controllo è quanto di più banale possa esistere: ad un punto preciso dell'azione bisogna spostare il joystick in una della quattro direzioni ortogonali o premere fuoco. Se si è scelta la giusta azione, si passa alla successiva sequenza animata, altrimenti si "muore".

Ovviamente anche la scelta del momento in cui impartire il comando è critico: una mossa corretta al momento sbagliato comporta un disastro certo. E' da notare che si riceve una vita supplementare solo dopo 10000 punti.

Il gioco è distribuito su quattro dischetti, ma anche possedendo due drives viene comunque utilizzato il solo drive interno. Non è necessario comunque un grande scambio di dischetti durante la partita vera e propria.

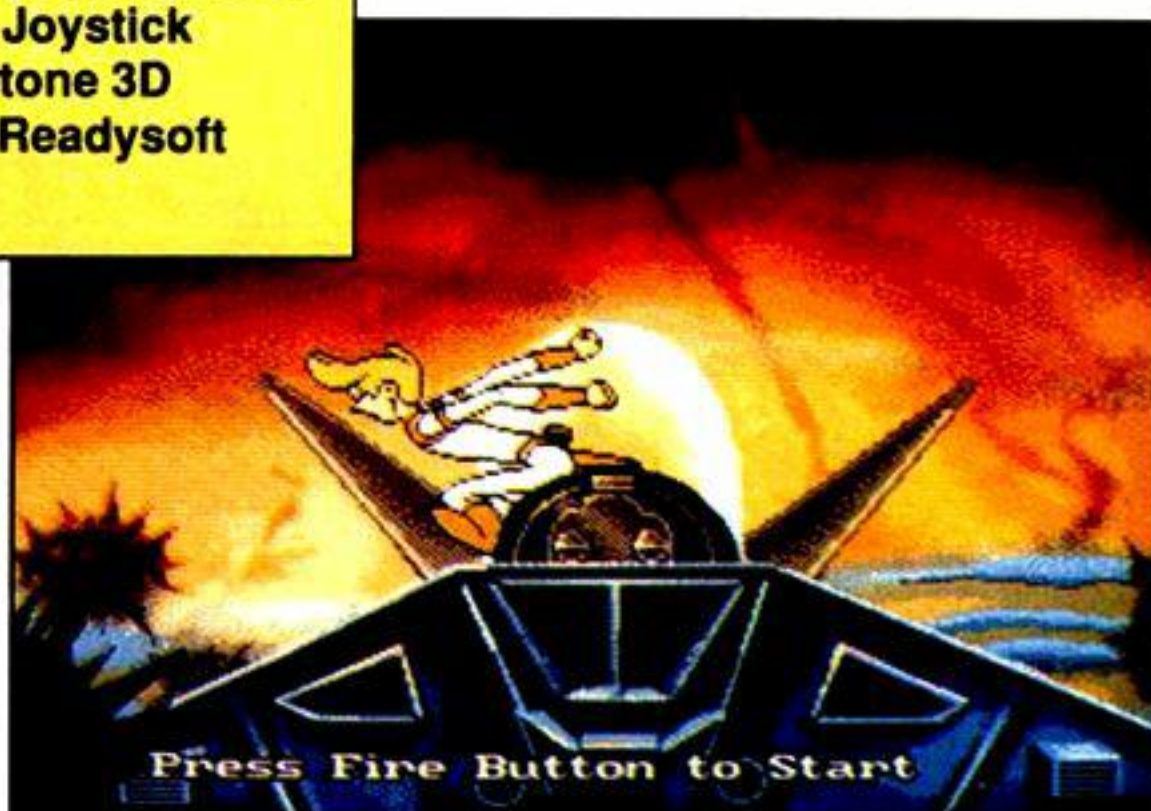
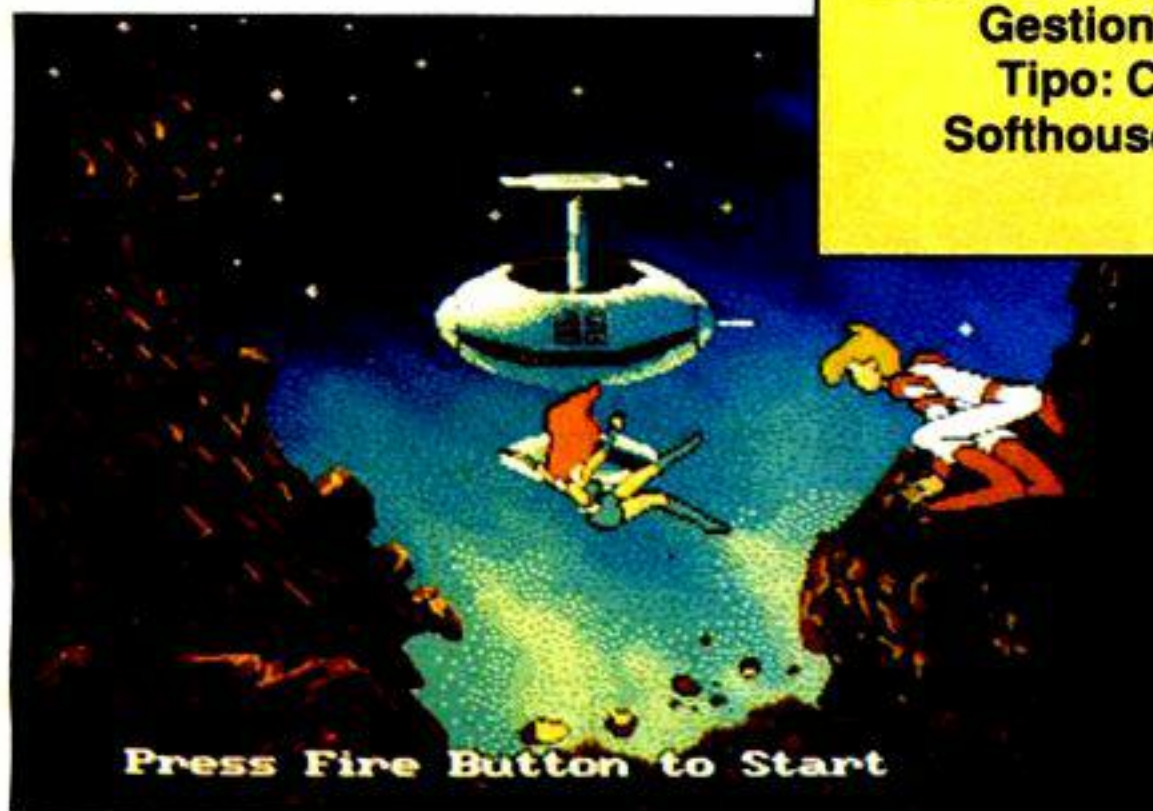
Tecnica

La grafica è ai massimi livelli concepibili nel settore delle animazioni (cinquanta fotogrammi al secondo) stile cartone animato di Walt Disney. Il numero di scene è molto alto e le animazioni sono appassionanti ed incredibili per realismo. L'interazione è comunque, come sempre, ridottissima: premi un pulsante, muovi il joystick e vinci o perdi. Del resto, con tutta questa grafica, sarebbe troppo chiedere un gioco concettualmente poco più che banalissimo.

Voto

Space Ace è come Dragon's Lair: un dimostrativo grafico superlativo, dotato di scarsissimi incentivi e certamente completabile una volta sola, dopo la quale il suo enorme fascino iniziale si dissolve. 6-.

Un eccellente cartone animato, ma niente di più

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Cartone 3D
Softhouse: Readysoft

QUARTZ

Gli alieni rompono le scatole veramente dappertutto: se non sono nello spazio o all'orizzonte, devono addirittura andare a livello sub-atomico nei reticoli cristallini molecolari.

In questa ultima produzione di *Paul "Spindizzy" Shirley*, autore di vari videogiochi classici per C/64 e Spectrum tra i quali ricordiamo anche *Confuzion* ed *Asteroids*, la classica trama di uno "spara spara" è sceneggiata infatti faccia a faccia con gli atomi.

La prima fase di Quartz, che è anche la più grande, si svolge all'interno della struttura di un cristallo di quarzo, con la nostra navicella armata per trasformare gli atomi di idrogeno (rappresentati come sfere gialle e rosse) in *quark*, a loro volta da colpire per trasformarli finalmente in *neutrini*.

E' necessario distruggere tutti i nuclei prima che l'agitazione termica ne provochi l'esplosione, con la conseguente distruzione della nostra navicella. La fase è insomma molto simile a quella di giochi come *Meteoroids* od *Asteroids*, con grafica ovviamente molto più animata e colorata, e l'astronave che può procedere in otto diverse direzioni.

Quando si è ottenuto un numero sufficiente di neutrini per riempire i tre contenitori colorati (rosso, giallo e blu) sulla destra dello schermo è possibile scegliere un'arma supplementare da installare a bordo della nostra navicella: granate, vari tipi di cannone, scudi protettivi, nuovi contenitori di armi eccetera.

Da qui è anche possibile memorizzare lo stato attuale di gioco su disco per riprenderlo eventualmente in seguito.

La fase dura un certo tempo, dopo il quale si entra nel nucleo e ci si trova a giocare con possibilità di movimento solo ortogonale, nuovi "alieni" rappresentati da altri atomi, ostacoli al suolo (?) da superare eccetera.

Se si riesce a sopravvivere ed a distruggere comunque il nucleo, si ritorna alla fase precedente dopo essere passato alla interfase di scelta di armi extra. La scenografia comunque cambia ancora: globuli, anticorpi e cuori pulsanti fanno da scenografia sino all'arrivo del classico guardiano di fine livello.

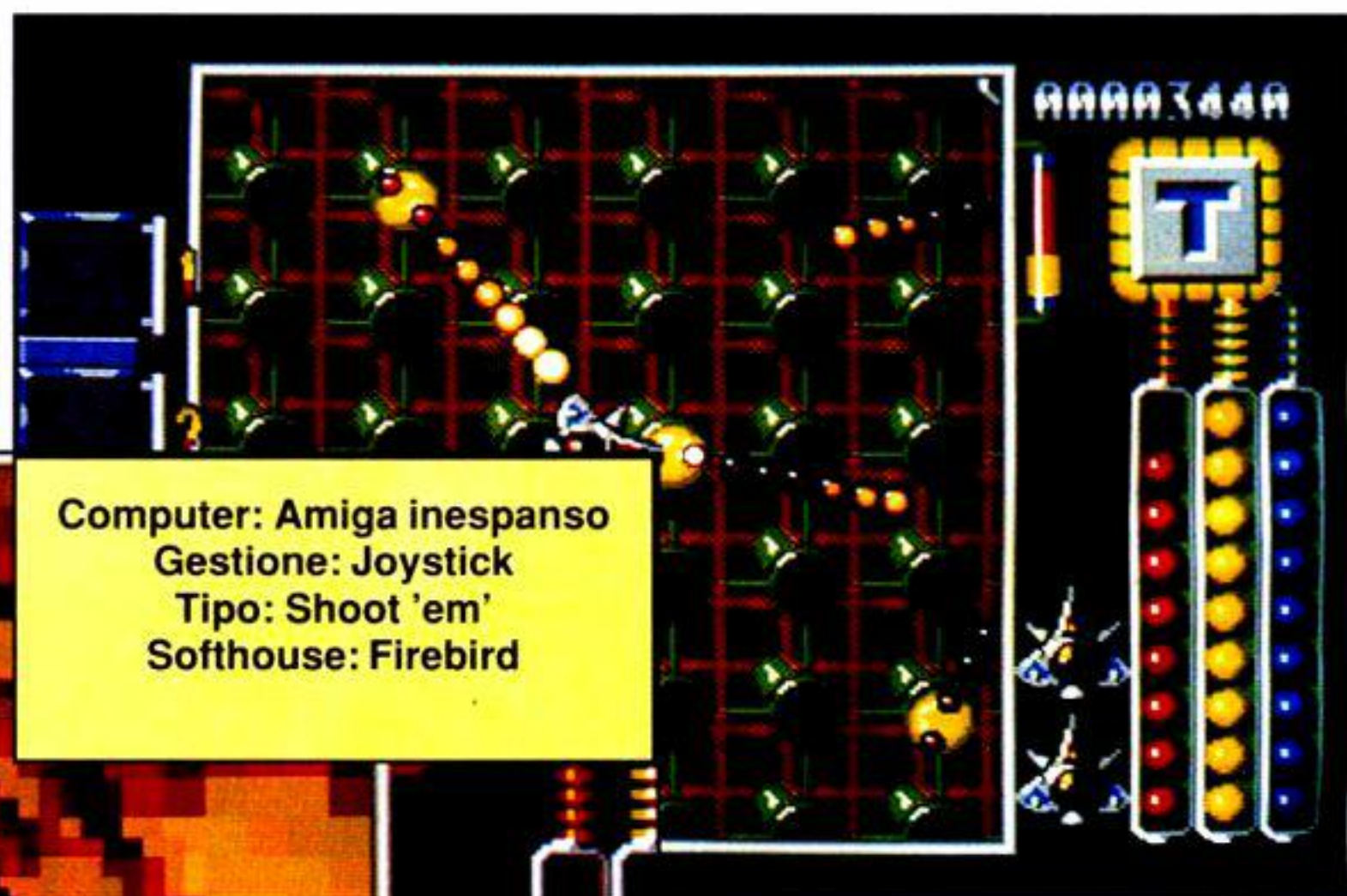
Il ciclo si ripete per un numero imprecisato di volte.

Forse nessuno ha ancora terminato Quartz, nè può dire esattamente di quanti livelli è composto. In ogni caso, da quanto abbiamo visto, in ogni fase cambia il tipo di scorrimento (a volte in otto direzioni, a volte solo in tre), gli sprites e la scenografia.

Tecnica

Grafica eccellente. E' stato chiaramente usato il sistema di animazione a doppia pagina (non chiedeteci come facciamo a saperlo), ovvero ciò che si vede sullo schermo è stato disegnato un attimo prima in *RAM* e poi spostato e visualizzato materialmente in un batter d'occhio via *Blitter*.

Ciò consente una grande nitidezza e fluidità dei movimenti, in quanto è presumibile che la velocità di aggiornamento dello schermo raggiunga i 50 fotogrammi al secondo, come nei cartoni animati di migliore qualità.

Lo sfondo è coloratissimo e spesso confonde la vista al giocatore, ma questo è probabilmente voluto per rendere la vita più difficile al giocatore. Gli effetti sonori sono veramente buoni, a tratti sorprendenti.

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Shoot 'em'
Softhouse: Firebird

Voto

Carino, giocabile, ben realizzato, non richiede fatica per comprendere il suo funzionamento. 7 1/2.

X-OUT

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Shoot 'em' up
Softhouse: Rainbow Arts

La campagna pubblicataria estera descrive *X-Out* come un *assalto multisensoriale*, ma in effetti il programma è un classico "spara spara" tipo *Denaris* o *Katakis*, come quelli prodotti generalmente dalla *Rainbow Arts*:
Una collezione di idee già viste, miscelate sapientemente con una realizzazione tecnica di qualità superiore alla media.

X-Out si svolge in scorrimento orizzontale da sinistra verso destra (in gergo si dice *sideway scrolling*) e si basa su quaranta tipi differenti di alieni, otto livelli, larghi ciascuno circa venticinque volte lo schermo video, *interfase* di compravendita armi, grafica a 48 colori (con una tecnica software particolare e non standard) ed animazioni a 50 fotogrammi al secondo.

L'ambientazione è nel profondo degli abissi dove gli alieni hanno impiantato dei negozi. Il fondo marino viene quindi usato come base di partenza per lanciare attacchi ai terrestri sulla terraferma.

Si inizia con 12000 *crediti* nel negozio di armi, dove si possono acquistare armi di varia potenza, missili, laser, armi supplementari ed anche altre navicelle, in modo simile a quanto visto in *Xenon II*. Poi lo svolgimento è il classico "spara a tutto ciò che si muove", con notevole dose di fantasia visibile negli sprites alieni animati (alcuni, però, assomigliano a quelli di Xenon II...).

Tecnica

L'introduzione animata è scenosissima, il sonoro veramente eccellente, sia per quanto riguarda la musica, sia per quanto riguarda gli effetti sonori puri e semplici. La grafica di animazione è realizzata con 50 fotogrammi al secondo, raggiungendo la massima fluidità apprezzabile dall'occhio umano. La grafica di scenografia è coloratissima e fantasiosa, grazie all'uso di uno speciale modo "retinato" di grafica che consente di ottenere 48 colori apparenti sullo schermo conteporaneamente invece dei 32 massimi consentiti normalmente senza ricorrere allo *HAM*. In particolare, i guardiani di fine livello sembrano partoriti dalla mente di un programmatore ubriaco di birra avariata. Basti pensare che alla fine del primo livello si trova una specie di mostro con la testa a teschio di cavallo ed il corpo a metà tra un polipo ed una formica.

Voto

Nulla di originalissimo, ma certamente realizzato in modo ineccepibile. 7 1/2.

SWITCHBLADE

Sebbene sia venduto col marchio *Gremlin's*, questo gioco è effettivamente stato prodotto dalla *Core Designs*, già autrice di *Rick Dangerous*, distribuito dalla *Firebird*. Buon sangue non mente, comunque, in quanto quel programma fu un buon successo sia nella versione Coin Op sia nella versione Amiga, ed anche *Switchblade* si presenta con le carte in regola per piacere a molti.

Il protagonista qui è *Hiro*, un indomito combattente giapponese dotato di braccio cibernetico programmabile, che vive su *Traxx*, invaso dal malvagio *Havok* con i suoi terribili scagnozzi. Hiro deve recuperare i sedici pezzi della *Spada di Fuoco*, simbolo della sua casta, e uccidere Havok per vendicarsi.

La scena di gioco è costituita da cinque livelli sotterranei, dove i pezzi da raccogliere sono disseminati casualmente, come in tutti i giochi a piattaforme che si rispettino. Le guardie Havik sono di differenti gradi di aggressività e pericolosità, ovvero di mobilità e grado di energia sottratta a Hiro quando lo colpiscono.

Per colpire Hiro si preme il tasto di fuoco: una barra in basso sullo schermo cresce, sinchè non lo si rilascia, ed il colpo portato dal nostro eroe cambia concordemente. Rilasciando il pulsante ad 1/4 Hiro sferra un pugno, a metà potenza (e sino a 3/4) provoca un calcio alto, mentre a piena potenza sferra un calcio basso. Colpendo e spostando a destra contemporaneamente si possono abbattere più avversari con un colpo solo.

Come di consueto, vi sono degli oggetti che, raccolti, forniscono potenza ad Hiro, nonchè armi extra, utilizzabili contro i nemici più ostici, come ad esempio palle di fuoco, *Shuriken* (stelle di acciaio appuntite) e dardi.

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade a piattaforme
Softhouse: Gremlin's

Tutte le armi hanno comunque un'autonomia limitata, quindi Hiro deve affidarsi soprattutto alle proprie forze.

Una caratteristica che ricorda molto "Rick Dangerous" è che molti schermi sono effettivamente accessibili soltanto colpendo i macigni sul muro, altrimenti rimangono nascosti. Hiro, infatti, può vedere soltanto la porzione di scena visualizzata; quindi può capitare di scendere ignorando parecchie stanze solo perchè non le si è scoperte "manualmente".

Tecnica

Gli effetti sonori stereofonici sono eccellenti, e di grande atmosfera.

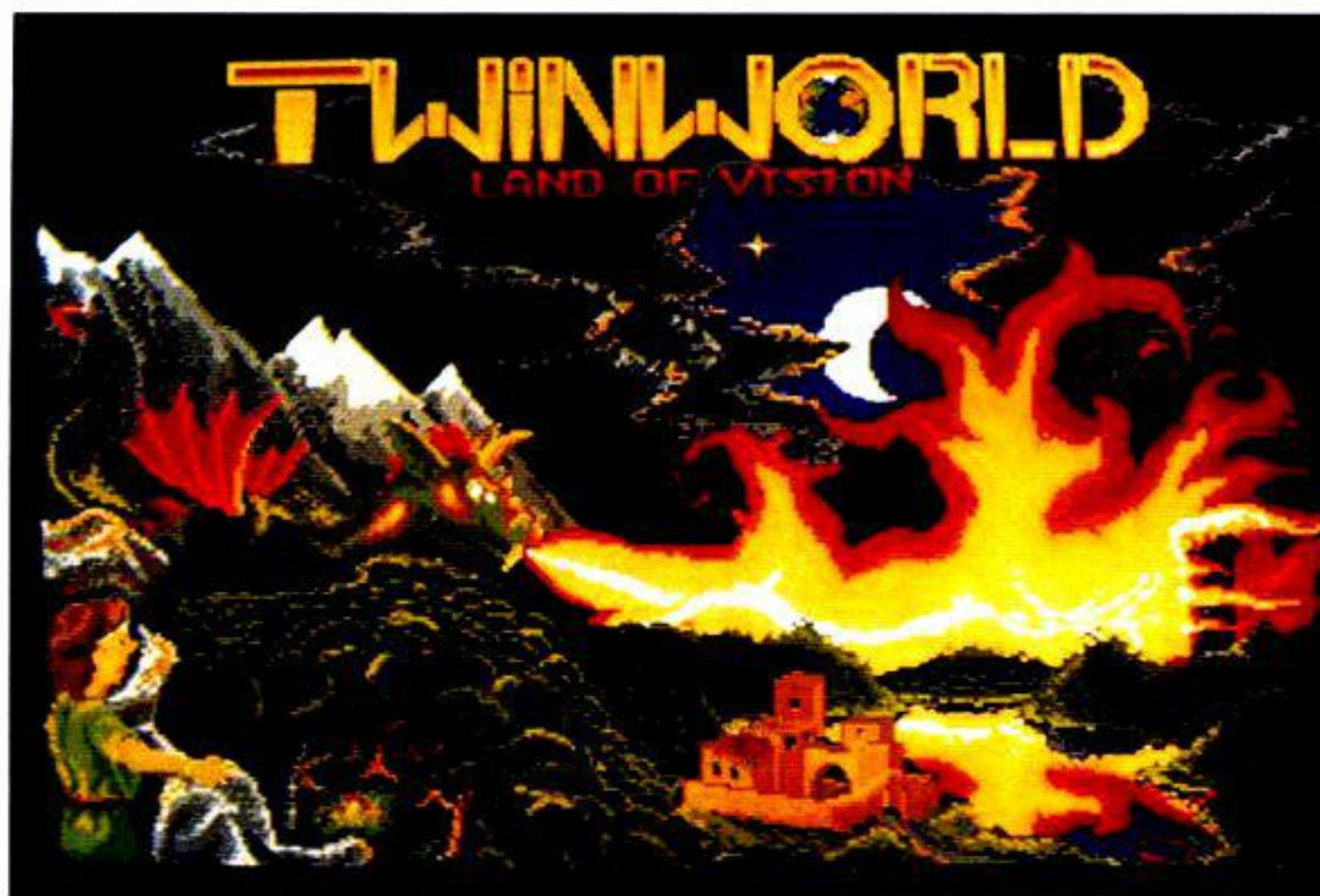
La scena è rigidamente bidimensionale, le scene sono semplici e ben disegnate, con animazioni degli sprites ineccepibili, anche perchè sono piuttosto piccole. Unica pecca, la sequenza di salto, certamente non la migliore che ci sia capitata di vedere.

Avremmo preferito forse una maggiore varietà di colpi nel repertorio di Hiro, dal momento che le ridotte dimensioni degli sprites non avrebbero richiesto molto sforzo da parte dei programmatori.

Voto

Giocabile senza troppe difficoltà, tecnicamente valido e con buoni incentivi. 7+.

TWIN WORLD

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade a piattaforme
Softhouse: Ubisoft

I *Gaspars* erano una razza prospera e pacifica, protetta dalla magia buona della famiglia *Carikens*, sinchè l'immancabile cattivone, *Maldur*, non rubò l'amuleto che donava loro le facoltà magiche. Il cattivone era però incapace di gestire l'amuleto incantanto, che così esplose in ventitrè pezzi che rimasero disseminati per tutto il paese.

Il gioco è stato scritto dagli stessi autori francesi che idearono *Super Mario Bros*, ed è fondamentalmente un gioco a piattaforme.

Ciascun livello è composto da un mondo superiore ed uno inferiore, con vie di accesso rappresentate da caverne intercomunicanti.

Per accedere al livello successivo bisogna comunque collezionare i pezzi dell'amuleto e trovare l'uscita di livello nascosta, ma identificavile dal simbolo disegnato sull'uscio.

Dal momento che Maldur non ha alcuna intenzione di lasciarvi ricostruire l'amuleto, ha infestato i vari livelli di mostriciattoli ed insidie molto pittoresche: cose indescrivibili che volano, strisciano e camminano (se così si può dire, visto che non sempre hanno gambe e piedi...). Per difenderci abbiamo tre tipi di sfere esplosive di varia potenza (una rimbalzante). Talvolta, disintegrando un mostro si possono poi raccogliere, dalle sue ceneri, delle munizioni.

Durante il gioco si scoprono altri bonus e gadget utili. Ad esempio un paracadute che consente di cadere da piattaforme sopraelevate senza accopparsi, e molle che consentono lo spunto indispensabile per attraversare certi crepacci o dislivelli particolarmente insidiosi. Per attivare questi *optional* è necessario premere un tasto qualunque della tastiera.

Tecnica

Lo sprite del giocatore è molto bello, rifinito (ha persino la riga tra i capelli), e animato in modo impeccabile. Tutta la grafica di movimento e di scenografia è veramente molto colorata e curata, senza mai sbavature o pecche. Gli effetti sonori sono divertenti, così come la musichetta di accompagnamento, disattivabile.

Voto

Un bel gioco di piattaforma, con vari incentivi, tecnicamente molto buono. 7 1/2.

DR. DOOM'S REVENGE

I personaggi dei noti fumetti americani sono ben conosciuti anche in Italia da parecchi anni. Qui sono sprites che saltellano e scazzottano per salvare il mondo sullo schermo di Amiga.

Il gioco

Il Dottor *Destino* ha rubato all'esercito USA un missile balistico intercontinentale a testata nucleare *ICBM*.

Per levare le castagne dal fuoco il governo ha chiamato due eroi senza macchia e senza paura: *Capitan America*, armato solo di scudo e muscolazzi da culturista senza ormoni, e l'*Uomo Ragno*, dotato sovrumanamente della superforza e superviscosità alle pareti, proprie dei ragni.

I due supereroi (marchio registrato) si precipitano a *Latveria*, dove vive il cattivone racchiuso in una futuristica armatura computerizzata. I due comunque non lavorano contemporaneamente, ma a turno, tra un intermezzo e l'altro, tra un robot assassino ed una bestiaccia creata in laboratorio...

Comunque il gioco è un normale spaccagrugni simile a molti altri, dove l'innovazione principale sta negli sprites ultrafamosi di questi personaggi che imperversano sui fumetti di tutto il mondo da circa quarant'anni.

La tecnica

Nessuna particolarità tecnica, se non gli sprites molto colorati ed i fondali, piuttosto suggestivi anche se non dettagliatissimi. Il movimento dei nemici dei supereroi non è quanto di meglio abbiamo visto sinora, mentre gli effetti sonori sono ridotti.

Più interessante l'incoraggiante inserimento di un'opzione che consente di "fare parlare" una tra quattro lingue differenti, tra le quali spicca anche l'italiano (almeno, nella versione importata e distribuita regolarmente in Italia, capito pirati?), anche se mi guarderei bene dall'usarlo per insegnare la grammatica e la sintassi ai bambini...

Computer: Amiga inespanso
Gestione: Joystick
Tipo: Arcade multischermo
Softhouse: Paragon's

Capitan America, il Dottor Destino e l'Uomo Ragno vivono nuove avventure sui vostri schermi

Il voto

Nulla di eccezionale, sotto tutti i punti di vista. 6 meno.

LA POSTA DEL C/128

a cura di Domenico Pavone

☞ 1 MONITOR PER 2 COMPUTER

✎ **Sono in procinto di acquistare un Amiga 500, ma, almeno per ora, non ho intenzione di disfarmi del mio buon C/128. Vorrei sapere se, acquistando anche un monitor (per ora uso un TV), potrò poi collegarvi tutti e due i computer, senza dovere ogni volta collegare e scollegare i cavetti vari.**
(Carlo Conti - Milano).

La cosa è possibilissima, a patto che il modello del monitor consenta tanto un ingresso videocomposito, quanto uno in RGB.

Il Commodore 1084, per esempio, può andare bene.

Occorre però prestare attenzione ad un dettaglio. Data l'abbondanza di accessi, è in pratica possibile collegare in permanenza l'Amiga alla presa Scart del monitor, il video 40 colonne del C/128 all'ingresso videocomposito, e il video 80 colonne dello stesso all'ingresso Rgb di tipo Din.

Quest'ultimo collegamento, però, interferisce con il segnale proveniente dall'Amiga, che adopera anch'esso l'Rgb.

Il che, schematicamente, comporta:

1) Se i due computer vengono usati non in contemporanea, non esiste problema di sorta: ogni modalità video è consentita.

2) Se C/128 ed Amiga sono entrambi in funzione, l'eventuale collegamento per le 80 colonne del C/128 deve essere disinserito, magari semplicemente sconnettendo lo spinotto sul retro del computer.

3) Se il C/128 è attivo solo sulle 40 colonne, basterà agire sul pulsantino selettore presente sul frontale del monitor per passare tranquillamente dal 128 all'Amiga.

Un'avvertenza: se, digitando qualcosa, sembra che non accada niente, non si ricorra immediatamente ad un cardiotonico.

Prima, è meglio sincerarsi di avere usato la tastiera giusta...

☞ SYS ALLA ROVESCIA

✎ **Può, una routine in linguaggio macchina, attivare un programma basic del C/128, come avviene per il caso contrario (Sys da Basic)?**
(Saverio Parrinello - Salerno)

Sì, anche se non proprio con le stesse modalità. Nel senso che, da basic, è possibile richiamare una routine in linguaggio macchina tramite Sys.

Quest'ultima, se si conclude con un doveroso RTS, dopo la sua esecuzione restituisce il controllo delle operazioni al programma basic, che può anche continuare a svolgere ulteriori compiti.

Nel caso inverso, una routine LM può lanciare un programma basic residente in memoria semplicemente invocando la routine di *Run*, che nel C/128 è allocata all'indirizzo $5AA6 (dec. 23206) di banco 15. Dopo averla richiamata, però, anche se viene adoperata l'istruzione JSR e non JMP, l'unico modo per rientrare nella routine LM sarà una Sys del programma basic.

Ma vediamolo in pratica.

Prima di tutto, un breve esperimento per verificare quanto detto: si resetti il

computer, quindi si digiti una riga basic tipo...

10 print "mimmo": rem basta con pippo!

...con la Rem del tutto facoltativa (non l'avreste mai pensato, vero?).

Ora, dopo il Return d'obbligo, si entri in *Monitor* (tasto F8) e si digiti direttamente *J F5AA6* (e Return), con **F** ad indicare il banco 15, e **J** il comando Jump, che eseguirà la routine in $5AA6.

Sullo schermo apparirà la stringa *Mimmo*, a testimoniare l'avvenuto Run del miniprogramma, ma non ci si troverà più in ambiente monitor.

Proviamo ora con una vera routine, anche se banale.

Senza spegnere o resettare il computer, si riattivi il monitor e, dopo aver digitato A 1300, si copi di seguito questa breve routine (per i meno esperti: il Return dopo ogni riga inserirà automaticamente gli indirizzi):

1300	LDA	#$00
1302	STA	$D021
1305	JSR	$5AA6
1308	LDA	#$00
130A	STA	$D020
130D	RTS,	

Con le prime due righe, si imposta il colore nero per lo sfondo; il JSR è un salto alla routine di Run, mentre le ultime due righe dovrebbero far assumere il nero come colore del bordo.

Si esca dal monitor con **X**, e si impartisca quindi, in modo diretto (da programma sarebbe la stessa cosa), *SYS 4864*. Lo sfondo diventerà nero, la stringa verrà stampata (il programma basic è ancora in memoria), ma il colore del bordo non sarà minimamente interessato dalle nostre manipolazioni. Dopo la routine di Run, insomma, si rientra irreversibilmente in basic, anche se il JSR farebbe pensare il contrario. Per eseguire anche il resto della routine, sarà necessario una ulteriore Sys 4872, che consentirà l'esecuzione delle istruzioni da $1308 in poi (nero anche per il bordo).

☞ UNA POKE FACILE FACILE

Qual è la Poke per abilitare il reverse sul C/128, e dove si trova il buffer della tastiera?

(Cristiano Pellegrini - Monza)

Il reverse, per chi è allergico al più tradizionale...

Print Chr$ (18) "stringa da stampare"

...è anche attivabile inserendo qualunque valore diverso da zero nella locazione 243 ($F3 in esadecimale).

La routine di sistema *Bsout*, che si occupa della stampa su video, testa questa locazione e, se non la trova azzerata, aggiunge 128 ai codici di schermo dei caratteri in output.

Il Return fa tornare le condizioni di default (reverse off), come pure il ben noto *Ctrl + 0* (zero), *Escape+O* (lettera O) e una doppia pressione del tasto *Escape*.

Si badi, quindi, che impartire da sola la poke in modo diretto non è produttivo: il Return ne annullerebbe subito l'efficacia.

Se proprio lo si vuol fare, occorrerà qualcosa come...

Poke 243, 1: Print "stringa"

...sulla stessa riga di comando.

Il suo uso, come ovvio, è più utile nell'ambito di un programma, soprattutto se Assembly.

Quanto al buffer di tastiera, nel C/128 occupa le dieci locazioni da 842 a 851 ($34A - $353), mentre in 208 ($D0) è memorizzato il numero di caratteri che lo impegnano.

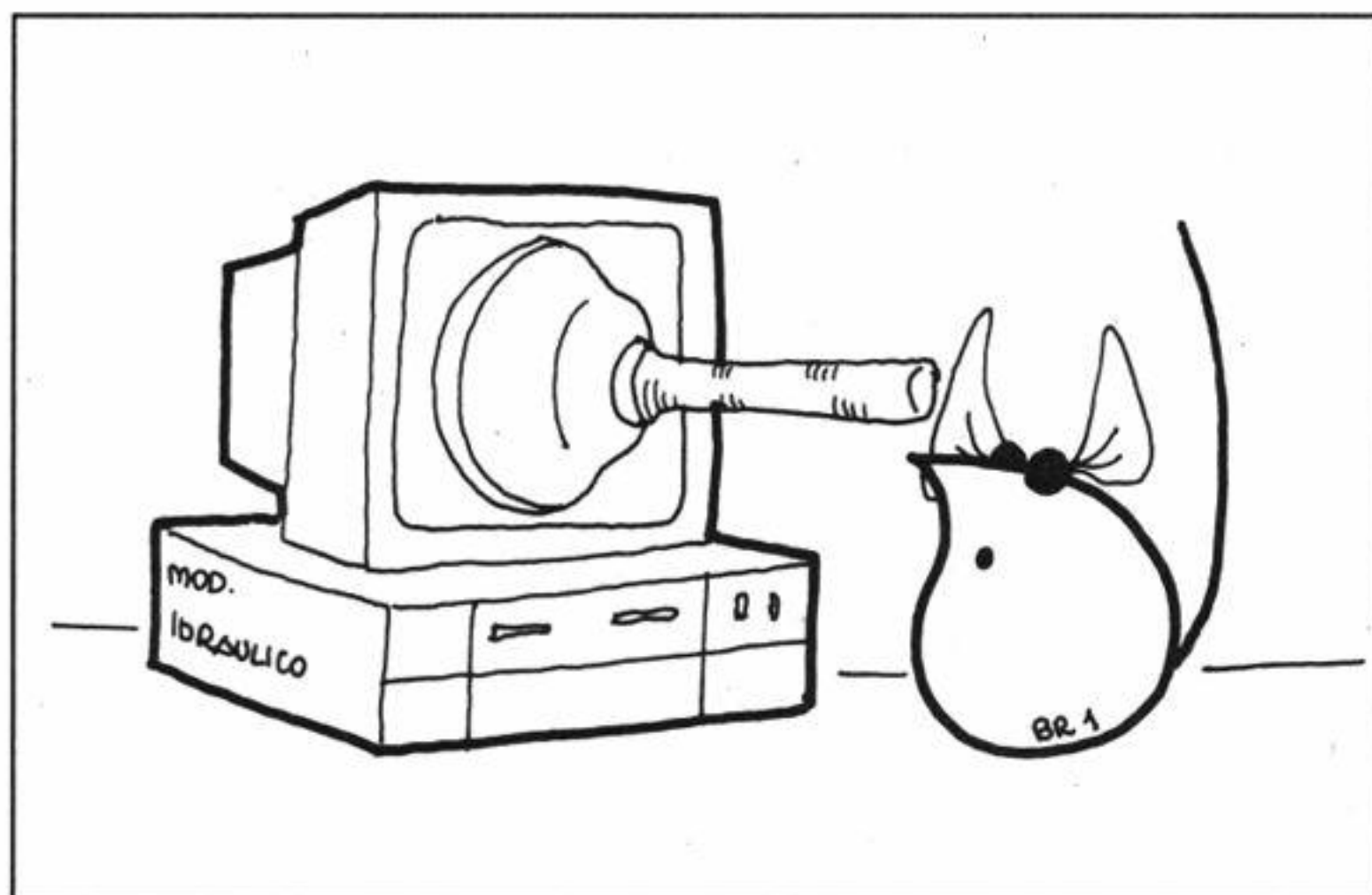
Alla gestione del buffer partecipa anche un'altra cella di memoria, la 2592 ($A20), il cui contenuto determina il numero massimo di caratteri che possono "sostarvi"; il valore presente in 2592 può essere modificato, ma senza superare la soglia di 10. Le applicazioni che questa particolare area di memoria consente sono state affrontate più volte nella rivista (Data maker, loader di programmi, ecc.): per maggiori dettagli, rimandiamo quindi il nostro lettore (e tutti gli interessati) ai relativi articoli.

☞ DA MACROASSEMBLER A WORD PROCESSOR

Ho imparato qualcosa sul linguaggio macchina seguendo i vostri articoli dedicati al C/64, e, come da voi consigliato, adopero il Macro Assembler Commodore anche per redigere i miei programmi per il 128. Vorrei sapere se è possibile, in qualche modo, digitare il testo (sorgente) di un programma con il word processor Superscript128, che non conosco molto bene. Inoltre, ho provato a caricare nel Superscript un sorgente fatto con l'editor del Macro Assembler: non ci sono problemi, solo che non compaiono i numeri di linea e il testo è tutto in minuscolo.

Si può ovviare in qualche modo senza dover modificare queste caratteristiche riga per riga?

(Paolo P. - San Donato)

```
10 rem trasforma un sorgente del macroassembler
15 rem commodore in file numerato e maiusolo
20 rem per il superscript 128
25 :
30 input "sorgente da trattare"; a$
35 fast: open 5, 8, 5, a$: if ds <> 0 then 105
40 if len (a$) > 12 then a$ = left$ (a$, 12)
45 open 6, 8, 6, a$ + ".num,s,w": if ds <> 0 then 105
50 gosub 80
55 get #5, a$: if st = 64 then 105
60 a = asc (a$)
65 if a > 64 and a < 91 then a = a + 128: a$ = chr$ (a)
70 print #6, a$;: if a = 13 then gosub 80
75 goto 55
80 nr = nr+1
85 if nr < 10 then nr$ = "00" + right$ (str$ (nr), 1): goto 100
90 if nr > = 10 and nr < 100 then nr$ = "0" + right$ (str$ (nr), 2)
95 if nr > = 100 then nr$ = right$ (str$ (nr), 3)
100 print #6, nr$ + chr$ (32);: return
105 print ds$: close 5: close 6: slow: end
```

Alla prima domanda la risposta è negativa, in quanto il word processor, oltre a quanto visibile, inserisce nel testo propri caratteri di formattazione (fine documento, ecc.) che l'assembler non può riconoscere, mentre, di contro, non troverebbe alcuni "propri" riscontri.

La possibilità inversa è invece di facile attuazione. Purchè un word processor agisca su file di tipo sequenziale, qualunque sorgente del Mac. Ass. può tranquillamente essere caricato e stampato tramite l'elaboratore di testi.

Nel caso particolare, Superscript (sia 128 che 64) come pure l'Easy Script del C/64 non pongono problemi di sorta, se non quelli lamentati dal lettore.

I numeri di riga del disassemblato non possono esserci, in quanto non vengono affatto inseriti dall'editor del Macro Assembler 64.

Inoltre tutti i caratteri alfabetici saranno in minuscolo.

Se, però, il file non è finalizzato all'assemblaggio, ma solo alla stampa o all'inserimento in documenti preesistenti, il problema è facilmente aggirabile: basta un programmino basic come quello proposto in queste pagine, che prelevi il file del Macro Assembler, operi le modifiche desiderate, e poi salvi il tutto in un altro file.

Per chi è interessato, la routine, una volta lanciata, chiede il nome del file sequenziale da trattare (p. es. *Sorg*), e ne crea uno con suffisso *.num* (p.es. *Sorg.Num*), troncando il nome originale se supera i 12 caratteri.

Il nuovo file, caricato da Superscript (o Easy Script 64), risulterà numerato come quelli prodotti da una hard copy del Macro Assembler, e con i caratteri alfabetici in maiuscolo.

+ FINESTRE

Come si fa ad attivare il comando basic Window in linguaggio macchina?
(Luca Barberi - Torino)

Per chi smanetta con l'assembler c'è più di un modo per definire una finestra video, e gli stessi metodi sono praticamente adoperabili anche da basic (se proprio il comando *Window* fa antipatia).

Anzitutto esiste una routine di sistema, guarda caso chiamata Window, locata all'indirizzo $CA1B (dec. 51739).

Questa, come consuetudine, va richiamata con una istruzione *Jsr* dopo aver settato opportunamente i registri *Accumulatore* ed *X*, nonchè il *Carry*.

Se quest'ultimo è azzerato (dall'istruzione assembly *Clc*, che sta per CLear Carry), i registri A ed X faranno riferimento alle coordinate dell'angolo in alto a sinistra della finestra.

Se il carry è settato (istruzione *Sec* = Set Carry), viene definito l'angolo inferiore destro.

L'accumulatore deve contenere la riga, compresa tra 0 e 24, mentre X specificherà la colonna (tra 0 e 39 oppure 0 - 79, a seconda dello schermo attivo).

Come esempio, ecco una tipica routine di definizione di una finestra:

1300	CLC		Angolo sup.sin.
1301	LDA	#$0A	Riga 10
1303	LDX	#$05	Colonna 5
1305	JSR	$CA1B	Routine Window
1308	SEC		Angolo inf.des.
1309	LDA	#$14	Riga 20
130B	LDX	#$1E	Colonna 30
130D	JSR	$CA1B	Routine Window
1310	RTS		Return

Per chi non ama le "raffinatezze" dell'Assembly, è disponibile un'altra scelta: settare direttamente le quattro locazioni da $E4 (dec. 228) ad $E7 (dec. 231), che determinano i limiti della finestra attiva; in particolare:

$E4	= Riga inferiore.
$E5	= Riga superiore.
$E6	= Colonna sinistra.
$E7	= Colonna destra.

Tutto ciò che occorre fare, in questo caso, è ricorrere a qualche *Poke* da basic, ovvero *Lda... Sta* in Assembly, e i nuovi limiti dello schermo saranno subito attivi.

Ma non è ancora finita.

Volendo, è anche possibile, tramite la stracitata routine di Output $FFD2, inviare un codice Escape (27) seguito dall'Ascii di *"T"* (84), dopo aver posizionato il cursore nelle coordinate dell'angolo superiore sinistro della finestra da impostare. L'operazione va eventualmente ripetuta per l'angolo inferiore destro, sostituendo il codice ascii della lettera **B** (66) a quello della **T**.

Chi preferisse quest'ultima (più laboriosa) procedura, può documentarsi sulle routine necessarie consultando le tabelle pubblicate negli ultimi numeri della rivista. Per resettare la finestra a tutto schermo, non è necessario reimpostare tutti i parametri: basta un semplice JSR $CA24, ovvero un salto (anche con Sys da basic) alla subroutine con indirizzo decimale 51748, e tutto tornerà alle condizioni di default.

STOP AL REGISTRATORE

Un listato brevissimo consente di esaminare da vicino una periferica obsoleta

di Dario Pistella

Viste le molte richieste, arrivate in Redazione, di programmi per tutti coloro che ancora non posseggono quella meravigliosa periferica chiamata *Drive*(!), e visto che durante il trascorso periodo natalizio saranno stati numerosi gli acquirenti di C/64 + registratore, abbiamo pensato di presentare un programma che sia, una volta tanto, utilizzabile solo dai possessori di tale periferica. Il breve listato presentato in queste pagine si occupa di un argomento fondamentale riguardante il *Datasette*.

Per registrare, verificare, caricare dati da cassetta, infatti, il registratore richiede al computer una tensione di alimentazione pari a 5 volts per permettere il movimento del motorino e l'attivazione della testina di lettura / scrittura. Pertanto ogniqualvolta vengono premuti i tasti *Play, Rewind, F. fwd*, il computer fornisce all'unità un voltaggio extra per lo svolgimento delle operazioni richieste.

Se, ora, analizziamo la locazione **1** della memoria del C/64, ci accorgiamo che, mentre normalmente assume il valore *55 ($37)* all'accensione della macchina, dopo la pressione di uno dei tasti del registratore, che richiedano l'invio di voltaggio extra, assume un valore diverso: *7*. Non appena il tasto premuto viene rilasciato, ritorna ad assumere il suo valore iniziale, cioe' *55*. Il programma proposto è in grado di bloccare il motorino del registratore quando lo si desidera tramite la manipolazione delle locazioni 1, di cui abbiamo già parlato, e 0, che permette di fissare, o meno, il valore desiderato all'interno della locazione 1, che altrimenti riprenderebbe immediatamente il valore corretto, facendo svanire l'effetto del comando impartito.

Il programma inizialmente sblocca il motorino, permettendo di riportarlo a 000 giri, quindi, dopo la pressione del tasto *Return*, lo blocca chiedendo di premere il tasto *F.fwd* e di digitare il "giro" sul quale si desidera che il motorino si blocchi.

In seguito darà alla locazione 1 il valore 7, permettendo quindi di attivare la periferica. Nella variabile *A* sarà presente un numero ottenuto tramite un algoritmo, valido per le più comuni cassette da 60 minuti, che il computer utilizzerà per decidere quando fermare il motorino.

Verrà infatti incrementata la variabile K, fino a quando i valori di K e di A non siano uguali. A questo punto il valore 55 verrà di nuovo forzato nella locazione 1 ed il calcolatore chiederà di premere il tasto *Stop* sul registratore, quindi rilascera' il motorino per permettere un eventuale caricamento. Il programma vuole essere sono un esemplificazione dei concetti fin qui esposti e si presta pertanto a modifiche di ogni genere. Ricordiamo inoltre che l'algoritmo è valido soltanto per le comuni cassette C/60 (Sony, BASF, Memorex, ecc.), con circa 410 - 417 giri di bobina. Si potranno pertanto avere errori di circa 2 - 3 giri rispetto al numero richiesto, data la variabilità dei giri di bobina di ciascun tipo di cassetta. Sta a voi trovare l'algoritmo specifico per la marca e per la durata delle cassette a cui siete interessati.

```
10 print chr$ (147) "riavvolgi";
15 print "completamente la cassetta";
20 print "e premi return": print
25 poke 0, 47
30 get a$:if a$ <> chr$  (13) then 30
32 gosub 1000
40 poke 0, 47
50 k = k + 1: if k  <> a then 50
60 poke 0, 55: poke 1, 55: print
70 print "ho effettuato" gi " giri."
75 print "premi STOP sul registratore"
77 poke 198, 0: wait 198, 1: poke 0, 47: end
80 rem subroutine controllo motore
1000 poke 0, 55: poke 1, 55
1010 print "premi il tasto F.FWD";
1015 print "sul registratore...";
1020 print: input "quanti giri:"; gi
1030 k = 0: a = gi * 35: a = a + 35 * int (gi / 26)
1035 if gi > 60 then a = a - (7 * int (gi / 60))
1040 return
```

E' bene precisare che il numero di giri realmente effettuato può non essere in accordo con quanto visualizzato grazie alla variabile *Gi*.

Non bisogna dimenticare, infatti, che operiamo in *Basic* e che questo, appunto, è un linguaggio *interprete*. Se, ad esempio, inseriamo nelle prime righe numerose altre linee di *Rem* (che, apparentemente, non svolgono alcuna funzione), le stesse righe "devono" essere interpretate egualmente prima che l'interprete decida di non utilizzarne il contenuto.

La riga 50 contiene il comando *GoTo 50* che, per essere interpretato, costringe l'interprete a rintracciare la riga 50. Per svoilgere questo banale compito, il Basic esamina, una per una, tutte le righe del programma a cominciare dalla prima disponibile (la *10*, nel listato presentato). Se, quindi, prima di arrivare alla fatidica riga 50 ne incontra un numero elevato (anche se contenenti solo Rem), dedica molto tempo alla loro selezione e ne può conseguire un ritardo sensibile.

Se non ci credete, provate ad inserire un congruo numero di righe prima della 50; poi, fateci sapere...

COLLABORARE CON C.C.C.

IN ATTESA DI GIUDIZIO

Se desiderate avere un giudizio sui programmi che avete scritto, è indispensabile inviarli su disco (***non*** ci è possibile esaminare lavori inviati su nastro o, peggio, su carta) ed inserite, nella busta, una descrizione chiara ed esauriente sulle funzioni svolte dal programma inviato. In alternativa potete inviare, come "commento", un nastro cassetta su cui avete inciso, servendovi di un normale registratore corredato di microfono, le vostre osservazioni sull'utilizzo del programma inviato. In questo caso cercate di non superare i cinque minuti di commento; inoltre, durante la registrazione, ponetevi dinanzi al computer e descrivete le operazioni da compiere non tralasciando di riferirsi ad alcuni esempi pratici che possano meglio illustrare le caratteristiche del programma inviato.

La rivista *Commodore Computer Club* offre l'opportunità, a tutti i suoi lettori, di stabilire un "contatto" che permetta di approfondire la conoscenza del mondo Commodore. Sono molti, infatti, gli utenti volti alla perenne ricerca di tecniche nuove, procedure insolite o applicazioni particolari.

Altrettanto numerosi, inoltre, sono i lettori che desiderano un giudizio sui programmi che hanno scritto con tanta pazienza e forza di volontà. Nè possiamo trascurare gli utenti che, appassionati di videogames, si dilettano a sproteggere giochi, individuare le Poke più strane o, semplicemente, a copiare dischetti applicando tecniche inedite.

Considerando le innumerevoli attività svolte da altri utenti (utilizzo ottimale di pacchetti applicativi, realizzazioni di piccoli accessori hardware, sviluppo di comunicazioni, di procedure in *Basic*, in *Assembly*, in *C*) è inevitabile che la nostra rivista dedichi ampio spazio per soddisfare la curiosità di tutti i suoi lettori.

A tale scopo sono state create vere e proprie rubriche, cui ci si può rivolgere con fiducia, a patto di rispettare scrupolosamente le note riportate in queste pagine.

SENZA RESTITUZIONE

Gli articoli, i dischi, le realizzazioni hardware e tutto il materiale inviato alla Systems editoriale non verrà restituito neanche in caso di mancato utilizzo del materiale stesso.

LA POSTA

La rubrica "*La Posta*", presente in tutti i periodici che si rispettino, è andata sempre più specializzandosi a causa delle notevoli diversità degli argomenti trattabili.

Allo scopo di sveltire lo spoglio della corrispondenza, consigliamo, pertanto, di indicare chiaramente, nell'intestazione della lettera, l'argomento richiesto.

Nel caso in cui si desiderasse rivolgere più domande, è indispensabile che ogni domanda sia trascritta su un foglio di carta.

Evitate la trattazione di più argomenti su un unico foglio e non siate prolissi: esponete il problema con parole semplici e, soprattutto, siate brevi!

Nel caso in cui ciascuna domanda dovesse superare la lunghezza di dieci righe dattiloscritte, è indispensabile inviare la domanda stessa su disco, e non su carta.

Le rubriche "specializzate" sono ormai numerose: *Posta 64, Posta 128, Amiga, Drive 1541/71, Amigassebly, Amiga Basic, Macroassembler 64, Software professionale*, eccetera.

COLLABORAZIONE "ATTIVA"

Se ritenete di aver scritto un programma interessante e desiderate che sia pubblicato, telefonate in Redazione (tel. 02/52.49.211, meglio se al pomeriggio) per concordarne la pubblicazione. Qualsiasi programma che dovesse pervenire senza un preventivo accordo rischierebbe, infatti, di essere messo da parte, in attesa di un (improbabile) esame che verrà effettuato esclusivamente dopo la verifica dei lavori inviati dagli altri lettori.

ARGOMENTI

Ricordiamo ai lettori che è possibile richiedere argomenti legati esclusivamente al "mondo" Amiga, C/64 oppure C/128.

Non è possibile affrontare argomenti legati ad elaboratori obsoleti (cioè Commodore serie Pet, Vic 20, C/16, Plus/4, C/128 modo CP/M).

Considerando la "tendenza" attuale dell'utenza, si consiglia, tuttavia, di esaminare con attenzione l'eventualità di abbandonare il C/64-128 per "passare" al mondo Amiga oppure, in alternativa, al sistema Ms-Dos.

POSTAMIGA

a cura di Domenico Pavone

☞ TASTI ALIENI

✎ **Possiedo un Amiga 500 con tastiera italiana da circa due anni, e non sono ancora riuscito a capire a che cosa servono i simboli a forma di freccetta che appaiono sotto alcuni pulsanti del tastierino numerico.**
(Stefano Beccani - Genova)

In ambiente strettamente Amiga, in pratica non servono a niente.

Alcune tastiere di Pc compatibili, sprovviste di tasti riservati allo scopo, adoperano, per gli spostamenti del cursore, proprio quelli del tastierino numerico contrassegnati con una freccia, che indica la direzione del movimento.

Di conseguenza esistono programmi, sempre dedicati all'Ms - Dos, che prevedono l'uso dei tasti in questione per portare a spasso il cursore sullo schermo.

Con (rara?) lungimiranza, è stata quindi prevista la possibilità che simili programmi, grazie al supporto di emulatori software oppure di schede XT / AT, possano capitare tra le mani di qualche "amigo" con propensioni Msdossiane.

☞ START IMPURO

✎ **Mi sono organizzato un dischetto di lavoro personalizzato seguendo le indicazioni dell'articolo da voi pubblicato sul n. 68 (*Workbench lavori in corso*). Ho copiato i files necessari al sistema dal disco Workbench 1.3 adoperando l'utility *Climate*, e sono molto soddisfatto del risultato, nel senso che tutto funziona perfettamente ed ho parecchio spazio nel disco. Ho però notato uno strano particolare, che non si verifica con il Workbench originale: quando lancio il sistema, prima che appaia lo schermo iniziale, mi viene segnalato un messaggio *Pure bit not set* senza altre conseguenze, almeno credo. Come mai?**
(Sante Prioli - Roma)

Il messaggio indica chiaramente che, durante l'esecuzione della startup - sequence, può essersi verificata una anomalia legata al comando *Resident*.

Questo, come specificato sul manuale del Dos in dotazione, richiede che un file da rendere residente in memoria sia "rientrante".

I files in possesso di tale caratteristica sono segnalati dal cosiddetto bit "pure" (puro), la cui presenza può essere verificata da *Shell* adoperando *List*.

Per esempio, con *List L* il file *Shell-Seg* contenuto in questa directory sarà caratterizzato dalla specifica *--p-rw- d*, con la **P** che indica appunto il bit Pure.

L'esempio non è scelto a caso, come non a caso si è prima detto che *può* essersi verificata un'anomalia.

Andando infatti a leggere la startup-sequence *(Type S / Startup - sequence)* del disco Workbench, si noterà la presenza di queste due righe...

Resident Cli L: Shell - Seg SYSTEM pure add

Resident c: Execute pure

In entrambe, al comando Resident è aggiunta l'opzione *Pure*, in presenza della quale il sistema rende in ogni caso il file residente, anche se il bit P non è settato, ma segnalando la cosa con il messaggio "pure bit not set".

Nel caso sia presente anche la specifica System, in effetti, il file non viene comunque inserito nella lista residente di Sistema, ma la procedura non abortisce.

Senza l'aggiunta dell'opzione Pure, se lo stato del file manca del relativo bit P, si ha invece una mancata esecuzione del comando, con relativa segnalazione del Dos ed interruzione di un eventuale batch file come la startup- sequence.

Sulla base di queste premesse, risulta evidente la causa degli "strani" messaggi appurati dal nostro lettore: uno (o più) file, come *Execute* (directory C) e *Shell-Seg* (directory L) non avevano il bit P settato. La cosa non provoca alcun inconveniente perchè Execute viene reso residente ugualmente, mentre Shell-Seg, che serve all'attivazione di Shell, quando necessario verrà in ogni caso rintracciato dal sistema sul dischetto (lo stesso varrebbe per Execute, se non fosse reso residente).

Quanto alla causa dell'anomalia, è presto detto: il programma *Climate* (almeno fino alla versione 1.2 x), effettivamente molto comodo quando si hanno molti file da manipolare, copia solo i bit di protezione *rwed*, ignorando quelli introdotti con il Dos 1.3.

Ove occorra, si può comunque porre rimedio adoperando, dopo la copia, il comando *Protect* (vedi manuale) sul file incriminato.

+ TROPPA FRETTA

✎ Ho l'impressione che il comando *Ed* non funzioni bene. E' già successo più di una volta che, dopo avere modificato la startup-sequence, uso il comando Escape + X per uscire dall'editor, ma, dopo avere resettato il sistema, appare la finestra Dos e tutto si blocca. Controllando bene, la startup-sequence mi risulta vuota e devo riscriverla daccapo.
(Stefano Garlata - Napoli)

✌ Il problema non è sicuramente da imputarsi all'editor Ed che, seppur poco brillante, il suo lavoro lo svolge egregiamente.

A giudicare dalla scarna descrizione degli avvenimenti, molto probabilmente, dopo Escape+X, il reset è impartito con troppa precipitazione.

Anche se la finestra dell'editor scompare, molto spesso la memorizzazione del file sul disco prosegue ancora per un certo lasso di tempo (questione di secondi, comunque).

Se il reset interrompe questa fase, il file (nel tuo caso, la startup-sequence) risulta poi *empty* ad un controllo successivo, e quindi non può svolgere la sua funzione.

Nel caso della startup-sequence, provocherà un arresto della procedura di start, anche se non un vero e proprio blocco: la finestra *Cli* sarà comunque attiva.

Per evitare simili conseguenze è sufficiente aspettare, prima di premere *Ctrl + A + A*, che la spia del drive si spenga.

In ogni caso, a meno che non si stesse editando un nuovo file, non è necessario riscrivere tutto, in quanto il precedente contenuto viene memorizzato da Ed nella directory **T** (se non c'è, la crea).

Quindi, se il forzato "abort" è intervenuto sulla startup- sequence, basterà impartire...

Copy T/ ed-backup to S/ startup-sequence

...per riavere disponibile il file, pronto per essere modificato da Ed.

Stavolta, però, con meno frenesia nel provarne l'efficacia!

☞ DA GRAFICA A GRAFICA

✎ Vorrei sapere se è possibile trasformare una schermata grafica in alta risoluzione interlacciata in una di risoluzione inferiore, per esempio 640 x 250, e se esiste un modo per disporre su Amiga di schermate create da computer Ibm compatibili (senza adoperare emulatori o schede).
(Franco Masinelli - Udine)

✌ La risposta è positiva per entrambi i quesiti.

Per modificare la risoluzione di una schermata, rapidamente e senza troppi passaggi, il modo più facile è ricorrere al programma *Transfer 24* della New Teck Inc., "appendice" naturale di *Digi Paint 3*.

Con questa utility basta settare i parametri di schermo desiderati, e caricare l'immagine da trasformare: al resto provvede lei.

Naturalmente, la videata va poi salvata su disco, ed è utilizzabile da qualunque programma gestisca lo standard Iff, *Show* e similari compresi.

Per di più, esiste anche una versione *Pal* di Transfer24, che consente di utilizzare la più estesa risoluzione orizzontale del video europeo, e di elaborare una immagine da NTSC (standard americano) al Pal, anche se qualche deformazione è inevitabile.

Per ciò che riguarda la possibilità di manipolare schermate provenienti dal mondo Ms-Dos, esistono nell'ambito del Pubblico Dominio parecchi programmi per Amiga che consentono di visualizzare lo standard più diffuso tra i Pc, il *GIF*.

Un paio di nomi: *Showgif* e *Virtgif*.

Inoltre, cosa alquanto più utile, i dati di una immagine in formato Gif possono essere rielaborati da programmi quali *Hamsharp*, sempre Public Domain, in modo da trasferirli in più amighevoli files IFF.

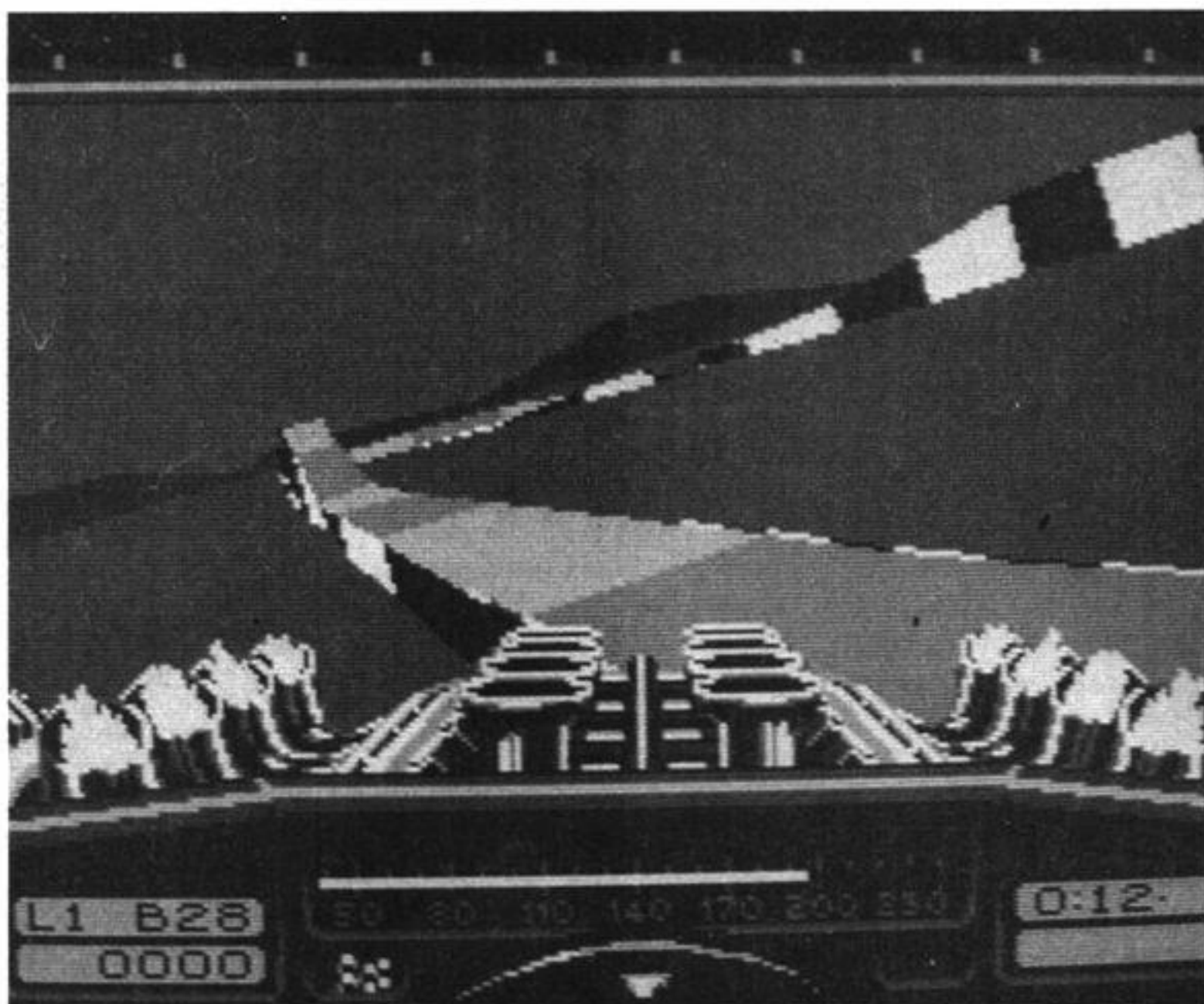
Ovviamente, prima di qualunque altra operazione, è indispensabile rendere accessibile ad Amiga il file grafico da visualizzare o modificare.

In altre parole, occorre "trasportare" il file da un disco Ms-Dos ad uno Amiga-Dos: adoperando p. es. il già citato *Dos2Dos*, oppure tramite connessione (diretta o via modem) tra computer.

☞ RIQUADRI ELASTICI

✎ Vorrei inserire, in un mio programma basic, la possibilità di delimitare una zona dello schermo con un riquadro modificabile col mouse, come si fa per il ridimensionamento delle normali finestre. Ho provato adoperando l'istruzione *Line* per disegnare il riquadro e poi, ad ogni movimento del mouse, ridisegnandolo col colore di fondo per cancellarlo e creandone un altro alla nuova posizione.

Funziona, ma si cancella anche il contenuto dello schermo.

C'è un modo per evitarlo?
(Rino C. - Avellino)

L'algoritmo generale per modificare "plasticamente" le dimensioni di un riquadro è senz'altro corretto.

Adoperando, però, solo le risorse dell'interprete Basic, l'unico metodo per evitare di cancellare, assieme al perimetro del riquadro, quello che ci sta "sotto", comporterebbe il ridisegnare ogni volta l'intero schermo.

C'è un'altra soluzione, decisamente più comoda e veloce, che richiede l'utilizzo di una sola funzione della libreria grafica, chiamata *SetDrMd* (che sta per Set Drawing Mode).

Per verificarne l'efficacia, si provi a digitare questa manciata di istruzioni...

```
LIBRARY "graphics.library"
RastPort& = WINDOW(8)
'----------------------------
start:
m = MOUSE (0): IF m <> -1 THEN start
x1 = MOUSE (3): y1 = MOUSE (4)
CALL setdrmd& (RastPort&, 2)
posizione:
x2 = MOUSE (1): y2 = MOUSE (2)
LINE (x1, y1) - (x2, y2), , b
WHILE MOUSE (0) = -1
IF x2 <> MOUSE (1) OR y2 <> MOUSE (2) THEN
   LINE (x1, y1) - (x2, y2), , b
GOTO posizione
END IF
WEND
CALL setdrmd& (RastPort&, 1)
LINE (x1, y1) - (x2, y2), , b
REM GOTO start
LIBRARY CLOSE : END
```

...badando, prima di lanciare il programma, che questo necessita della presenza del file *Graphics.bmap*, rintracciabile nella directory *BasicDemos* del disco *Extras 1.3* (oppure 1.2).

Come a molti noto, questo dovrà essere copiato nella directory *Libs* del disco utilizzato per lanciare il sistema, oppure, più semplicemente, andrà modificata la prima istruzione del programma in modo che contenga il *path* (percorso) del file. Se, per esempio, si è inserito Graphics.bmap in *Ram Disk*, la prima riga diventerà:

LIBRARY "Ram:graphics.library"

Se tutto è in regola, dopo il Run, basterà premere il pulsante sinistro del mouse e, mantenendolo abbassato, spostare il "topo" per regolare le dimensioni del riquadro che si andrà formando sullo schermo.

Rilasciando il pulsante, il quadrilatero assumerà il suo aspetto definitivo, tanto nella forma che nel colore, ed il programma si arresterà.

Eliminando la Rem della penultima riga, si può continuare a disegnare riquadri, sullo stile dei tool grafici più conosciuti.

Si badi, però, che è pur sempre il basic a gestire il tutto, con le sue imperfezioni; si eviti, perciò di eccedere in velocità tra il "click" e lo spostamento del mouse; potrebbe verificarsi qualche malfunzionamento (provvisorio).

Come ovvio, le poche righe qui riportate vanno poi adattate alle esigenze dei propri programmi.

Ma veniamo alla breve routine, il cui funzionamento generale può essere capito (dai meno esperti) semplicemente consultando il manuale del Basic.

Unico punto degno di nota, il già accennato uso della funzione *SetDrMd*, che determina la modalità di tracciamento della grafica.

Quando la si richiama (con *Call)*, necessita di due parametri:

1) L'indirizzo della struttura *Raster Port* della finestra corrente.

Ai non patiti del **C** o dell'*Assembly*, basti sapere che tale indirizzo è specificato dalla funzione basic *Window (8)* (si veda il manuale).

2) Un valore compreso tra 0 e 255. In particolare, con 1 si imposterà il tracciamento *Jam2*, corrispondente di solito al colore di primo piano.

Con 2, invece, viene effettuato un *Xor* (Or esclusivo) della sezione di schermo interessata, con il risultato che essa verrà ripristinata se (p.es.) una linea viene "spostata" altrove.

Come intuitivo, è quest'ultima peculiarità che consente alla nostra routine di funzionare correttamente.

In pratica, dopo aver determinato le attuali coordinate del mouse, viene impostata la modalità di tracciamento Xor, che rimane attiva in tutta la fase di aggiornamento della posizione (tramite *Mouse(1)* e *Mouse(2)*) e relativo disegno del riquadro. Dopo il rilascio del pulsante, viene invece reimpostata la modalità normale, ed il riquadro è ridisegnato un'ultima volta.

Stravolgendo il motivo di esistere della nostra routine, si provi anche ad eliminare (magari solo inserendo una Rem ad inizio riga) l'istruzione *Line* due righe ad di sotto dell'etichetta *Posizione*, e si proceda come prima descritto: l'utilità del programma ne risulterà compromessa... ma sarà l'occhio a goderne.

CAMPUS

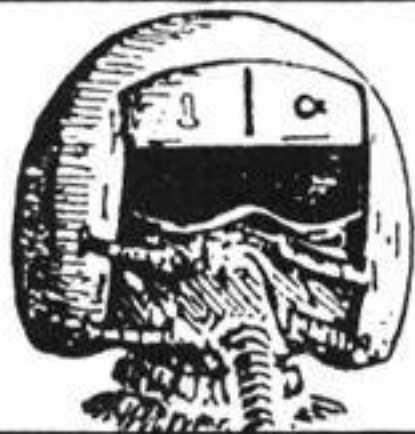
AMIGA

Due articoli (e vari programmi allegati) forse "duri" per i principianti, ma sicuramente apprezzati dall'utenza Amiga più evoluta.
La grafica, infatti, è il punto di forza di questo straordinario computer che spesso viene acquistato quasi esclusivamente per le sue eccellenti doti grafico-sonore.
In entrambi gli articoli si parla del "trattamento" di pagine grafiche; il primo può far parte della categoria "sprotezioni"; il secondo, invece, consente di manipolare immagini grafiche in ambiente AmigaBasic.

62 - ALLA RICERCA DELLA PAGINA PERDUTA

Volete esaminare con calma le schermate di un certo videogame particolarmente curato dal punto di vista grafico? Oppure volete capire qualcosa sull'allocazione delle pagine grafiche nella memoria di Amiga?
La procedura descritta nell'articolo consente, agli utenti smaliziati, di fare entrambe le cose. Attenzione, come al solito, a digitare correttamente i listati a corredo.

68 - VIDEO A RENDERE

L'interprete AmigaBasic, rintracciabile nel dischetto Extras, offre la gestione semplice di operazioni grafiche particolarmente complesse.
La generazione di schermate bellisime, tuttavia, si scontra con l'assenza di un comando specifico che permetta di registrarle su disco, per richiamarle in un secondo momento.
L'articolo, piuttosto lungo ma esauriente, descrive con semplicità il modo in cui Amiga gestisce le schermate ed illustra alcune procedure per risolvere il problema.
I brevissimi listati proposti, infatti, non sono altro che subroutine da personalizzare a piacimento ed inserire in qualsiasi nostro programma Basic destinato al trattamento delle immagini.

Dopo aver controllato con un apposito programmino l'esattezza dei DATA del primo file, potete passare, cari lettori, a questa seconda porzione della storia, in cui potremo seguire Primo ed amici mentre si cimentano nel loro allenamento al volo acrobatico ...

(da pagina 32)

GRAFICA AMIGA

ALLA RICERCA DELLA PAGINA PERDUTA

Il programma proposto questo mese permette di esplorare la memoria di Amiga alla ricerca di "parti" di schermate grafiche.

di Donato De Luca

Ecco un metodo per esaminare con calma le schermate grafiche di videogames, anche se "protetti"

Il motivo per cui ci riferiamo solo a "parti" di schermate, dipende dal fatto che una schermata grafica dell'Amiga può essere composta da più *BitPlanes* (piani bit).

Amiga può gestire un numero elevato di pagine grafiche; tuttavia si devono fare i conti con la memoria disponibile dal momento che 512k non sono poi così tanti, considerando che una singola schermata in low - res (320 x 200) a 32 colori occupa ben 40 kbytes.

Il sistema grafico di Amiga *non* è un sistema BitMapped; ciò significa che la sua pagina grafica non ha una locazione fissa, ma può essere posizionata a partire da una locazione qualsiasi dei primi 512 kbytes (la cosiddetta *Chip-Ram*). Tale limitazione è dovuta al fatto che i chip grafici dell'Amiga (e non soltanto loro) possono accedere solo ai primi 512 kb di memoria.

Per completezza ricordiamo che la restante memoria viene denominata *FastRam*.

La pagina grafica dell'Amiga può essere composta da un numero di BitPlanes variabile tra 1 e 6. Tuttavia il sesto BitPlane è usato quasi esclusivamente dai modi grafici speciali, cioè l'H.A.M. e l'HB.

A seconda del numero dei bitplanes utilizzati, si può avere un certo numero di colori presenti contemporaneamente sullo schermo.

Ovviamente un bitplane occupa memoria ed esattamente tanti bytes quanti ne misura la sua altezza per la sua larghezza divisa per 8.

Supponiamo di avere un bitplane in Low-Res (320 x 200); questo occuperà 8000 bytes:
320 x 200 / 8 = 8000

I bitplanes che compongono una schermata, come già detto, possono essere posti ovunque in memoria, anche se di solito vengono posti uno di seguito all'altro. Per gestire più di due colori, serve più di un bitplane; tuttavia nel programma proposto in queste pagine (ma solo nella versione C) viene visualizzato un solo bitplane.

Bitplanes	Colori
1	2
2	4
3	8
4	16
5	32

Il motivo di questa scelta, all'apparenza restrittiva, è che così facendo potremo vedere meglio i singoli bitplanes.

LA VERSIONE C

All'inizio del programma (versione C) vengono specificate alcune costanti. Successivamente, dopo aver creato un *array* (colori), inizia la *Main*, in cui si definiscono alcune vari-

abili; in seguito si apre la libreria *graphics.library*.

Di seguito si definiscono gli elementi per formare un display, cioè:

view
viewport
rastport
bitmap
colormap
rasinfo

Successivamente vi è il ciclo principale, che termina solo quando la funzione *Joy* restituisce un valore maggiore di 16; ciò accade quando viene premuto il pulsante fire della porta 2.

All'interno di questo ciclo, a seconda dei movimenti del joystick, viene scrollata la bitmap. In seguito si richiamano le funzioni della libreria grafica (*MakeVPort, MrgCop, LoadView*) le quali servono ad aggiornare il display.

Infine vi è *WaitTOF*: questa funzione attende che il pennello elettronico sia arrivato alla fine del quadro (cioè attende il *VBlank*). Quando il ciclo *while* termina, viene ripristinato il precedente display e, tramite la funzione *LiberaMemoria,* viene restituita al sistema la memoria utilizzata dal programma. Alla fine viene chiusa la libreria grafica.

LA VERSIONE BASIC

Questa versione è molto simile alla versione C perfino (ma solo in apparenza) per la velocità d'esecuzione. Nella versione in C si è utilizzata la funzione *WaitTOF* per sincronizzare il movimento della bitmap (o meglio dei puntatori alla bitmap) con il *VBlank*. Nel Basic, invece, ciò non è stato necessario perchè la sua lentezza sostituisce egregiamente la funzione WaitTOF.

Da notare, comunque, che mentre il modulo eseguibile della versione C occupa solo 4.5 k (circa) la versione Basic, anche se più breve, per funzionare necessita, ovviamente, della presenza dell'interprete Basic il quale è leggermente lungo (100 k e passa...).

Chi, quindi, possiede solo 512 kbyte di Ram, è molto probabile che, caricando l'interprete, quest'ultimo si possa sovrapporre ad una porzione di schermata grafica. Ovviamente il discorso potrebbe cambiare se compilate la versione Basic.

COME FARE

1 Caricate un gioco, o un qualsiasi programma che faccia uso di pagine grafiche.

2 Aspettate che venga visualizzata almeno una schermata grafica.

3 Resettate l'Amiga.

4 Caricate *Esploratore*.

5 Tramite il joystick (in porta 2), vi potrete "muovere" all'interno della chip-ram alla ricerca di un bitplane.

6 Quando avrete trovato un bitplane, continuate a muovervi nella stessa direzione impostata. Infatti è molto probabile che, di seguito, troverete anche gli altri bitplanes (ammesso che ve ne siano).

7 Premete il pulsante fire per uscire.

Due sono le versioni proposte:la prima in C, la seconda, addirittura, in Basic!

Chi non è pratico di linguaggi "avanzati" può limitarsi a digitare il programma in Basic

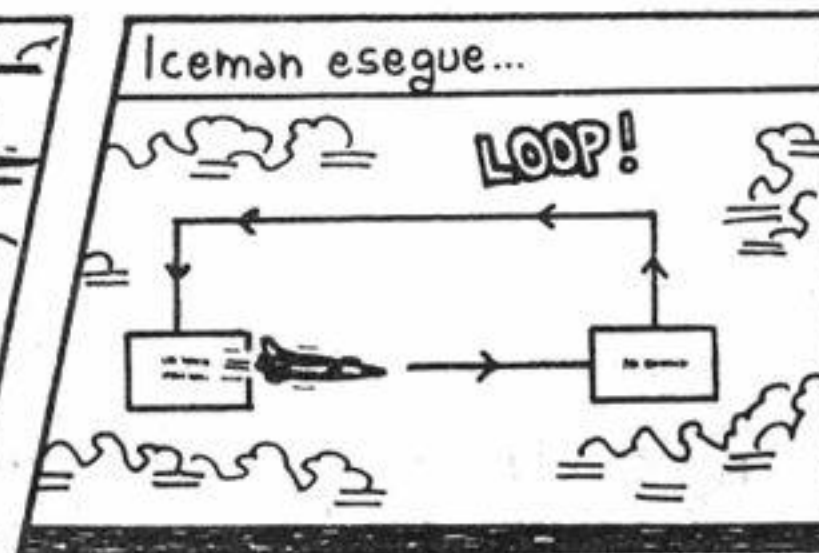
```
/* ESPLORATORE CHIP RAM */
/* WRITTEN BY */
/* DONATO DE LUCA */
/* */
#include <exec/types.h>
#include <graphics/gfx.h>
#include <hardware/dmabits.h>
#include <hardware/custom.h>
#include <graphics/gfxmacros.h>
#include <graphics/copper.h>
#include <graphics/view.h>
#include <graphics/rastport.h>
#include <graphics/regions.h> */
#include <graphics/clip.h>
#include <exec/exec.h>
#include <graphics/text.h>
#include <graphics/gfxbase.h>
/* #include <functions.h > */
#define NB 1L /* n. bitplanes */
#define LARG 320L /* larg. bitmap */
#define ALT 200L /* alt. bitmap */
#define VPLARG 320L /* larg. viewport*/
#define VPALT 200L /* alt. viewport */
#define MEMORIA_INSUFFICIENTE -1000
struct View view; /* strutture */
struct ViewPort viewport;/*view,viewport*/
struct ColorMap *cm;/* colormap */
struct RasInfo rasinfo;/* strutture */
struct BitMap bitmap;/* RasInfo BitMap */
struct RastPort rastport;/* RastPort */
extern struct ColorMap *GetColorMap();
struct GfxBase *GfxBase;
struct View *vecchio;
USHORT Colori[]={
 0x000 , 0x00f , 0x005 , 0x006 , 0x007 ,
 0x008 , 0x009 , 0x00a , 0x00b , 0x00c ,
 0x00d , 0x00e , 0x00f , 0x500 , 0x600 ,
 0x700 , 0x800 , 0x900 , 0xa00 , 0xb00 ,
 0xc00 , 0xd00 , 0xe00 , 0xf00 , 0x000 ,
 0x000 , 0x000 , 0x000 , 0x000 , 0x000 ,
 0x000 , 0x000
};
/* Vedere commento n. 1 */
UWORD *TabellaColori;
void _main() /* prg. principale */
{
/* definisco alcune variabili */
LONG i;
int a,x,y;
a=0;
/* apro la graphics.library */
GfxBase = (struct GfxBase *)
OpenLibrary("graphics.library",0L);
if (GfxBase == NULL) _exit(100);
/* no memoria? si-> fine prg. */
vecchio = GfxBase->ActiView;
/* salvo il vecchio schermo */
InitView(&view);
InitVPort(&viewport);
view.ViewPort = &viewport;
view.Modes = SPRITES;
viewport.Modes = SPRITES;
InitBitMap(&bitmap, NB , LARG , ALT );
/* vedere commento n.2 */
InitRastPort(&rastport);
rastport.BitMap = &bitmap;
rasinfo.BitMap = &bitmap; /* bitmap ass. */
rasinfo.RxOffset = 0;
/* x angolo superiore sinistro */
rasinfo.RyOffset = 0;
/* y angolo superiore sinistro */
rasinfo.Next = NULL;
viewport.DWidth = VPLARG;/* larg. viewport
*/
viewport.DHeight = VPALT;/* alt. viewport */
viewport.RasInfo = &rasinfo;
/* struttura rasinfo associata */
cm = GetColorMap(32L);
TabellaColori = (UWORD *)cm->ColorTable;
for(i=0; i<32; i++) {
```

COMPILAZIONE

La versione in C del programma può essere compilata sia con il compilatore Lattice che con Manx.

Nel caso si voglia usare Manx, si deve rendere effettiva la linea...

/ #include <functions.h> */*

...eliminando i caratteri di barra (/) e di asterisco (*).

```
*TabellaColori++ = Colori[i];
}
viewport.ColorMap = cm;
for(i=0; i<NB; i++) {
bitmap.Planes[i] = (PLANEPTR)
AllocRaster(LARG, ALT);
if(bitmap.Planes[i] == NULL) {
LiberaMemoria();
_exit(MEMORIA_INSUFFICIENTE);
}
}
/* vedere commento n. 3 */
/* XXXXXXXX */
while (a<16)
{
/* vedere commento n. 4 */
x=0;
y=0;
a=joy(); /* Richiamo la funzione Joy*/
if (a==1) { y=-1; }/*Joystick alto*/
if (a==2) { x=1; } /*Joystick destra*/
if (a==4) { y=1; }/*Joystick basso*/
if (a==8) { x=-1; }/*Joystick sinistra*/
rasinfo.RyOffset=rasinfo.RyOffset+y;
rasinfo.RxOffset=rasinfo.RxOffset+x;
MakeVPort(&view, &viewport);
MrgCop(&view);
LoadView(&view);
/* vedere commento n. 6 */
WaitTOF();
/* vedere commento n. 7 */
}
/* vedere commento n. 8 */
LoadView(vecchio);
/* Richiama vecchia schermata */
LiberaMemoria();
/* Restituisce la memoria */
CloseLibrary(GfxBase);
/* Chiude la libreria aperta */
}
/* Fine del programma principale */
LiberaMemoria()
{
LONG i;
/* memoria utilizzata
al sistema */
for (i=0; i<NB; i++) {
if (bitmap.Planes[i] != NULL)
{FreeRaster(bitmap.Planes[i], LARG, ALT);
}
}
if (cm != NULL) FreeColorMap(cm);
FreeVPortCopLists(&viewport);
FreeCprList(view.LOFCprList);
return(0);}
joy()
{
/* commento n. 9 */
int x;
short a,*b;
char *c;
b = 0xdff00c; /* porta 2 */
c = 0xbfe001; /* CIA-A */
x = *b;
a = 0;
x &= 0x00000303;
if(x & 0x0002) a |= 2;
if(x & 0x0200) a |= 8;

if((x & 0x0001)^
((x & 0x0002)>>1)) a |= 4;
if(((x & 0x0200)>>1)^
(x & 0x0100)) a |= 1;
x = (short)*c ;
if(!(x & 0x80)) a |= 16 ;
return(a);
}
```

Inoltre dovrete cancellare il segno di sottolineato (_)presente prima di *Main*.
Per compilare con Lattice, digitate...
LC ESPLORATORE.C
...e, per linkare con Blink,...
Blink Lib: C.o Esploratore.o Lib Lib: Lc.lib, lib: Amiga.lib
Con compilare con Manx, invece, dovrete...
Cc Esploratore.c
...e per linkare...
Ln Esploratore.o -lc32

Caricate un gioco e fate in modo di far apparire una videtata; ora basta resettare, caricare e lanciare uno dei listati di queste pagine

Le notizie relative all'uso corretto del compilatore Lattice C dovrebbereo esser già note ai nostri lettori; tuttavia riportiamo le fasi essenziali per compilare il programma

Di solito risulta abbastanza difficile digitare un programma, seppur abbastanza breve, senza commettere errori; inoltre la procedura per invocare il compilatore (esempio: Lattice C) ed il suo linker non è molto breve. Si consiglia, quindi, di creare un file contenete la suddetta procedura, in modo che ogni volta che vorrete compilare basterà far eseguire il file tramite il comando *Execute*.

Si sottolinea che, per ovvi motivi, il file sorgente si deve necessariamente chiamare *Esploratore.c.*

Volendolo denominare con un nome diverso, dovete digitare il suo nome seguito dal suffisso **.C** oppure da **.O** (a seconda se compilate o linkate) al posto di "*Esploratore*".

Durante la compilazione con Lattice compariranno alcuni warning:

Warning 30: pointer do not point to same object

Warning 30: pointers do not points to same object

Warning 85: function return value mismatch

Non dovete fare altro che ignorarli.

Si sconsiglia vivamente di compilare la versione C, se avete un solo drive.

Se fate partire la versione basic, assicuratevi che sul disco di sistema siano presenti i due file...

exec.bmap

graphics.bmap.

Li dovreste trovare nel disco *Extras*, nella directory *BasicDemos*. Nel caso li abbiate cancellati, li dovrete ricopiare dal disco basic originale.

Se, addirittura, li avete cancellati dal disco Basic originale (ma maritereste l'impiccaggione) li dovrete creare seguendo la procedura descritta su *C.C.C. n. 60* (pagina 72).

Riportiamo, comunque, la procedura da seguire:

1: Avviare l'Amiga con il disco Workbench.

2: Aprire la directory BasicDemos di Extras e lanciare il programma ConvertFD.

3: Rispondere con...

Extras: FD1.2/exec_lib.fd

...alla richiesta del file **.fd**.

4: Rispondere con...

nomevostrodisco: exec.bmap

alla richiesta del file **.bmap**.

5: Attendere finchè il programma non finisce di scrivere il file *exec.bmap.*

Ripetere di nuovo la procedura solo che, al posto di *exec*, dovete digitare *graphics*.

Se, per caso, non avete i file **.fd** oppure avete cancellato il programma convertitore *ConvertFD*, non vi resta che... piangere.

OSSERVAZIONI

Nel programma, per vari motivi, non sono stati inseriti i necessari commenti che qui di seguito, tuttavia, riportiamo.

Commento 1

In *Colori* sono contenuti i valori dei colori che più avanti (nel programma) saranno utilizzati.

Commento 2

Si crea una bitmap con profondità (= numero dei bitplanes) uguale a *NB*, larghezza uguale a *Larg* e altezza uguale ad *Alt*.

La missione prosegue. Seguendo i precisi comandi di Primo Giovedini, gli aerei eseguono ardite manovre acrobatiche rimanendo a pochi pixel di distanza l'uno dall'altro... Tuttavia, come in ogni evento di programmazione, il BUG è in agguato...!

```
REM ESPLORATORE CHIP RAM
REM BY
REM DONATO DE LUCA
REM APRO LE LIBRERIE
DECLARE FUNCTION OpenLibrary& LIBRARY
LIBRARY "exec.library"
LIBRARY "graphics.library"
GfxBase& = OpenLibrary&(SADD("graphics.library"+CHR$(0)),0)
REM RICAVO L'INDIRIZZO DELLA VIEW ATTIVA
view& = PEEKL(GfxBase&+34)
REM RICAVO L'INDIRIZZO DELLA VIEWPORT ATTUALE
viewport&=PEEKL(WINDOW(7)+46)+44
REM RICAVO L'INDIRIZZO DELLA STRUTTURA RASINFO
Rasinfo&=PEEKL(viewport&+36)
REM RICAVO L'INDIRIZZO DI RXOFFSET E RYOFFSET
RxOffset&=Rasinfo&+8
RyOffset&=Rasinfo&+10
REM SALVO I VALORI ATTUALI DI RXOFFSET E RYOFFSET
X=PEEKW(RxOffset&)
Y=PEEKW(RyOffset&)
CRAX=X
CRAY=Y
REM SALVO LE CARATTERISTICHE DELLO SCHERMO DEL WB
OLDMOD=PEEKW(viewport&+32)
REM PORTO IL WB IN BASSA RISOLUZIONE
POKEW viewport&+32,0
TRACLSTEPPAN=5
REM MAIN: RIPETE FINCHE' IL FIRE PORTA 2 E' PREMUTO
WHILE TRACLSTEPPAN<>0
REM CONTROLLO SE IL FIRE E' PREMUTO
    TRACLSTEPPAN=PEEKW(12574720&) AND 128

REM LETTURA MOVIMENTI JOYSTICK
 jx=STICK(2)
 jy=STICK(3)

REM AGGIORNO X E Y
 X=X+jx
 Y=Y+jy

REM SCROLLO LA BITMAP
 POKEW RxOffset&,X
 POKEW RyOffset&,Y

REM FACCIO RICALCOLARE IL DISPLAY
 CALL MakeVPort(view&,viewport&)
 CALL MrgCop(view&)
 CALL LoadView(view&)
WEND
REM RIPORTO LE COORDINATE DELLA BITMAP AI
REM LORO VALORI INIZIALI
POKEW RxOffset&,CRAX
POKEW RyOffset&,CRAY
REM RIPORTO LO SCHERMO WB AL SUO MODO INIZIALE
POKEW viewport&+32,OLDMOD
REM FACCIO RICALCOLARE IL DISPLAY
CALL MakeVPort(view&,viewport&)
CALL MrgCop(view&)
CALL LoadView(view&)
REM CHIUDO LA GRAPHICS.LIBRARY
CALL CloseLibrary(GfxBase&)
LIBRARY CLOSE
END
```

Usando il joystick ci si può spostare tra le "pagine" grafiche di Amiga

"Proteggere" un programma risulta, come intuitivo, una mera illusione; lo dimostra la procedura descritta che, in pratica, consente di esaminare con calma le schermate di un qualsiasi programma, anche protetto

Commento3

Si chiede al sistema operativo un'area di memoria per allocare la bitmap.

La sua grandezza dipende sia dalla risoluzione scelta (nel nostro caso bassa risoluzione, 320 x 200 pixels), sia dal numero di bitplanes utilizzati e quindi dal numero dei colori usati per la bitmap.

Poi vi è l'inizio del ciclo che finirà quando la funzione Joy restituirà un valore maggiore o uguale a 16; ciò avviene quando si preme il pulsante fire della porta 2. In questo caso, infatti, la funzione Joy riporta sempre valori maggiori o uguali a 16.

Commento4

Se provate ad eliminare le due righe successive (cioè x=0; y=0;) otterete un effetto abbastanza simpatico. In questo caso, tuttavia, dovrete rimmeterle fuori dal ciclo, al posto di...

/ XXXXXX */*

Ciò dipende dal fatto che le due righe in questione servono ad azzerare le variabili X e Y, in modo da evitare scrollate indesiderate.

Commento 5

Per avere uno scrolling più veloce bisogna modificare i valori assegnati ad x ed a y dagli IF. Provate ad assegnare i valori...

-10, 10, 10, -10

...in quest'ordine, e sentierete l'ebrezza della velocità...

Se, poi, siete proprio dei masochisti, per soddisfare i vostri desideri non avete altro da fare che mettere altri valori:

-30, 30, 30, -30

Bisognava usare il costrutto *Case* per la serie degli *If*, ma tanto...

Commento 6

Grazie allo scrolling hardware delle due righe precedenti viene scrollata la bitmap.

Succesimavente, tramite le funzioni di libreria *MakeVPort, MrgCop* e *LoadView* viene aggiornato il display.

Commento 7

Prima di ricominciare il ciclo, si attende l'inizio del Vertical-Blank onde evitare fastidiosi sfarfallii.

Commento 8

Se il loop *while* termina, vuol dire che il pulsante del fire è stato premuto ed il programma termina richiamando la vecchia schermata di partenza, restituendo al sistema la memoria che ha utilizzato e chiudendo la *graphics.library*.

La routine per la lettura del joystick è una routine di pubblico dominio.

VIDEO A RENDERE

Due tecniche basic per salvare, e riutilizzare in qualunque momento, le nostre migliori schermate

di Domenico Pavone

Le capacità grafiche di Amiga rappresentano il principale motivo di interesse tanto per i neo utenti di questa macchina, quanto per i suoi potenziali acquirenti.

Riusciranno, questi ultimi, a conquistare l'ambito oggetto di desiderio?

Ce lo auguriamo, ma non ci è dato saperlo.

Con maggiore cognizione di causa, possiamo però riferirci alle esigenze di chi il 16 bit non solo ce l'ha già, ma non si limita ad usarlo passivamente. Molti lettori (*Fabrizio Conti di Roma, Ezio Gamba di Asti, L. Vellioni*, ecc...), con diverse sfumature, ci hanno posto infatti lo stesso interrogativo, obbligandoci ad una risposta il più esauriente possibile.

Il quesito, più o meno, può essere riassunto in questi termini: data per scontata la superiorità tecnica degli svariati megatools in commercio, anche programmando in basic capita spesso di creare particolari schermate grafiche, talora per motivi estetici, ma più spesso per migliorare la leggibilità di un programma (istogrammi, testi ben formattati, ecc.).

La cosa, con *AmigaBasic*, non presenta difficoltà, data l'abbondanza di comandi dedicati alla generazione di figure, linee e tratteggi.

Se, però, si desidera salvare su disco (o Ram Disk) il risultato di tale fatica "digitatoria" (la schermata, insomma) bisogna fare tutto da soli, come pure per un (eventuale) successivo caricamento dell'immagine memorizzata su periferica.

Eppure, una tale risorsa potrebbe tornare utile in molte occasioni, per non parlare delle nuove prospettive che si aprono, una volta apprese le tecniche necessarie alla realizzazione pratica di tale "feature". Lancia in resta, dunque, e pronti all'azione.

INTRODUCTION

Intanto è bene precisare che non esiste un metodo unico per affrontare l'argomento, anche se poi i principi di base sono, in fondo, gli stessi.

Restando ben ancorati all'aspetto pratico, proviamo dunque ad accostarci per gradi alla soluzione del quesito proposto, realizzando due diverse tecniche per salvare e ricaricare uno schermo: una chiara e semplice da comprendere, tutta basic-basic *(listati 1 e 2)*, ed una più "tosta", con abbondante uso delle librerie di sistema, ma di contro molto più veloce ed efficiente della prima *(listati 3 e 4)*.

In ogni caso, per lasciare la più ampia libertà di scelta, sarà mantenuta una rigorosa compatibilità tra le diverse routine, come vedremo meglio tra breve.

Prima, però, è indispensabile qualche breve premessa.

Quando si parla di schermata, è infatti necessario intendersi sul significato concreto del termine. Senza stare a tergiversare, possiamo definire questa entità apparentemente astratta in termini di memoria, ovvero aree RAM che contengono tutti i dati necessari alla sua visualizzazione.

Più in concreto, una schermata è scindibile in tre elementi costitutivi:

Memorizzare le schermate grafiche di Amiga è ormai una procedura abbastanza semplice anche per un principiante

Amiga "depone" i dati relativi alle schermate grafiche in luoghi sempre diversi e difficilmente individuabili

▲ *I parametri che identificano il tipo di schermo.*
▲ *I parametri che ne specificano i colori.*
▲ *Le mappe di bit che rappresentano ogni singolo pixel dello schermo attivo.*

In effetti, avendo a che fare con Amiga, una simile schematizzazione è abbastanza riduttiva, ma, a parte alcuni approfondimenti che vedremo tra breve, è più che sufficiente per operare in Basic.

Per memorizzare su disco una schermata, sarà quindi necessario prelevare i dati delle tre componenti appena descritte, e passarli sequenzialmente (uno ad uno, per intenderci) ad un file preventivamente aperto in scrittura.

Come ovvio, per leggere (e visualizzare) il contenuto del file, occorrerà procedere con modalità opposte: aprire il file in lettura, prelevarne i singoli dati, e schiaffarli al posto giusto.

Già, ma qual è il *posto giusto* ?

Ed eccoci al nocciolo della questione.

Come spesso ribadito, *tutto* in un computer è da intendersi in termini di locazioni di memoria, ma pagando il giusto prezzo al multitasking di Amiga. In altre parole, queste locazioni non sono mai le stesse, o almeno non si può fare affidamento su una tale eventualità.

Quando, p. es., impartiamo da basic un comando *Screen*, è il sistema di Amiga a decidere dove andranno a finire i dati che lo riguardano, scegliendo tra le aree di memoria al momento non occupate da altre applicazioni (oppure da altri schermi!).

Per conoscere l'indirizzo di tali aree, si potrebbe aprire lo stesso schermo emulando in basic tecniche proprie di linguaggi come il C o l'Assembly (adoperando le librerie di sistema), ma complicando alquanto le cose, per di più inutilmente.

Perchè non sfruttare quanto Amigabasic già mette a disposizione, torchiando a fondo i suoi ben più semplici comandi?

Vediamo di rendercene conto in pratica, affrontando i problemi, a man a mano che sorgono.

PRIMI APPROCCI

Cominciamo, dunque, col dare un'occhiata ai listati 1 e 2, tutto sommato abbastanza brevi se sfoltiti delle numerose righe di commento.

Il listato 1, può essere suddiviso in due parti: la prima metà genera una schermata dimostrativa, la seconda la memorizza nella directory corrente con nome *pic*.

In testa al listato, sono poste le inizializzazioni, eventualmente modificabili.

Prime tra tutte le linee *Data* contenenti, ognuna, i tre parametri inerenti le *Palette* di colori che si intendono utilizzare (si veda il manuale).

A seguire, le quattro variabili da assegnare come parametri dell'istruzione Screen (*largh* = larghezza, *alt* = altezza, *prof* = profondità).

Si badi che, qualora si intendesse creare uno schermo con più colori modificando il parametro profondità, si renderebbe necessario aggiungere tante linee Data per quanti sono i colori, specificandoli tutti.

A meno, ovviamente, di non volere eliminare tutto ciò che riguarda il colore, tanto nella generazione della schermata, quanto nella sezione che si occupa di memorizzarla. Il numero di colori, dovrebbe esser noto, è espresso dalla potenza di due del parametro *Prof*(ondità). Piccola nota non del tutto inutile: il simbolo di elevamento a potenza, in alcune tastiere apparentemente assente, si ottiene pressando contemporaneamente i tasti *Shift* + *Alt* + *6* per la mappatura italiana, oppure *Shift* + *6* per quella americana.

Una volta creato lo schermo (istruzione Screen), è necessario aprirvi una finestra, nel nostro caso *Window 2*, a tutto video e senza barra del titolo (comunque liberamente ridefinibile). Il nostro Demo, per semplicità, opera in bassa risoluzione, e con soli quattro colori, letti dal ciclo *For... Next* subito dopo l'istruzione *Window*. A questo punto, ognuno è libero di sbizzarrirsi come meglio crede; nel listato, giusto per vedere in azione la routine, vengono creati alcuni cerchi concentrici con due segmenti rigati ai lati, ottenuti tramite i comandi *Area, Area Fill* e *Pattern*.

PROFONDO... NON SOLO ROSSO

Ciò che vediamo sul video, è organizzato in memoria in maniera che ad ogni pixel (non vorrete chiamarlo "puntino"?) corrisponde un bit in ogni bitplane.

Per capire meglio, si provi ad immaginare un quaderno (o un block notes, se preferite) con un limitato numero di fogli a quadretti.

Se consideriamo ogni singolo quadratino come un pixel, e quindi un bit, questo sarà ovviamente presente, nella medesima posizione, tanto nel primo foglio quanto in quelli sottostanti.

Ebbene, una *Bitmap* (il quaderno) altro non è che un certo numero di questi "fogli", tanti quanti specificato dal parametro *Profondità* nell'istruzione Screen, definiti con il termine BitPlane.

In ogni Bitplane (basta con fogli e foglietti!), sono contenuti tanti bit quanto richiesto dalle dimensioni dello schermo.

Nell'applicazione di queste pagine, p. es., avremo 2 Bitplane (profondità = 2), ognuno dei quali sarà costituito da 320 x 256 = 81920 bit.

E poichè ogni singola locazione di memoria rappresenta 1 byte (= 8 bit), in definitiva la dimensione del nostro bitplane può essere quantificata in 81920 / 8 = 10240 Byte, mentre, come ovvio, per valutare l'intera Bitmap occorrerà (in questo caso) raddoppiare tale valore.

Nei vari bitplane, alla condizione 0 / 1 che può assumere un bit, corrisponderà sullo schermo un pixel spento / acceso, nei diversi colori associati ad ogni bitplane.

Brillante scoop finale: in basic, adoperando l'istruzione Screen, il numero massimo di bitplane presenti in memoria sarà 5, limite massimo del parametro Profondità, per un totale di 32 colori.

Ricordate che, per visualizzare alcuni simboli, vi sono differenze tra una tastiera e l'altra

Per il salvataggio vero e proprio, non resta che usare *Open "pic"* per aprire in output un file con questo nome. Chiaro che, assieme al nome, può anche essere modificato il suo path, ovvero il posto dove si intende salvarlo.

Optando per la Ram Disk (molto più veloce), si potrà, per esempio inserire:

Open "ram: Pic" For Output As 1

Si badi che, per brevità, nei listati 1 e 2 non è stato inserito alcun controllo di errore negli accessi al file.

Se qualcosa non dovesse andare per il verso giusto, ci si ricordi, dopo un eventuale stop del basic, di chiudere schermo e finestra, magari digitando direttamente (nella finestra di output) *Goto fine* (e Return).

FILE A COLORI

Eccoci finalmente pronti per inserire nel file le già descritte tre componenti che caratterizzano la schermata. Le prime due, i parametri riguardanti schermo e colori, non pongono alcun problema.

A dispetto di quanto prima affermato, non bbiamo infatti alcun bisogno di andare a "rovistare" nella Ram di Amiga (e se soffrisse il solletico?).

Rispettando, infatti, il presupposto che la memorizzazione ed il successivo recupero di una schermata siano funzionali ad un programma basic, i parametri li abbiamo già belli e pronti. Per le caratteristiche di schermo, altro non sono che le variabili associate a larghezza, altezza, profondità e tipo: basterà trasformare i relativi valori in stringa, ed inviarli con *Print #1* al file prima aperto.

Per le prime due, dato che possono contenere un valore a due byte (un byte può assumere 255 come valore massimo), occorrerà adoperare *MKI$* (altra occhiata al manuale!), mentre per profondità e tipo basterà un *CHR$*, che occuperà un solo byte nel file.

Anche restando in "ambiente" Basic è possibile sfruttare le risorse del DOS

Stesso discorso per le tre componenti (rosso, verde e blu) di ogni Palette, già associate alle tre variabili con indice *r(x), g(x), b(x)* prima di "disegnare" la schermata. Qui, però, per schivare i mille trabocchetti legati ad una corretta lettura della mappa colore in memoria (alquanto complicata), si ricorre ad un artifizio.

Ogni componente colore, come noto, è rappresentata da un numero compreso tra 0 ed 1, quindi anche con valore frazionario.

Dovendo però salvare tale valore sotto forma di stringa, per poi poterlo recuperare con la maggiore semplicità possibile, l'ideale sarebbe quello di avere un numero intero, e dalle dimensioni il più ridotte possibili.

Un byte, risulterebbe proprio l'ideale.

Basta moltiplicare per 100 la variabile - colore, per farla sempre rientrare nel range 0 - 100, "a portata" di un Chr$ per essere memorizzata. Quando si vorrà recuperare la schermata (listato 2), una preventiva divisione per 100 rimetterà le cose a posto.

Nel listato (sezione *parametri colore*), un banalissimo ciclo *For... Next* provvede alla bisogna senza sprecare più di un paio di righe (occhio ai punti e virgola!).

TERRA IN VISTA!

Non rimane, infine, che memorizzare nel file *"pic"* l'ultimo elemento, la *Bitmap* (vedi riquadro). Stavolta, non c'è trucco che tenga: bisognerà per forza di cose tuffarsi nella memoria di Amiga e, dopo aver ricavato i necessari indirizzi (vedi secondo riquadro), operare a suon di *Peek* per leggere tutti i byte di ogni BitPlane, e "scaricarli" nel solito file.

Per fare un po' più in fretta, il loop che se ne occupa (l'ultimo, in fondo al listato) procede 4 bytes per volta *(Step 4)*, consentendo l'uso di *Peekl*, che per l'appunto legge quattro bytes *(L = Long)* dalla memoria.

Il valore ottenuto viene trasformato in stringa da *MKL$*, e memorizzato nel file.

Ad operazioni ultimate, il programma emette un *Beep*, dopo un lasso di tempo non proprio trascurabile (niente paura, si rimedierà tra poco). Sulla base di quanto visto, è ora intuitivo capire il meccanismo di azione del listato 2, che svolge il compito opposto: legge il file creato dal listato 1, e ne visualizza l'immagine.

Mandandolo in esecuzione, si potrà toccare con mano il formarsi dei due diversi Bitplane, anche qui in tempi non del tutto accettabili, abituati come si è alla velocità di Amiga. Schematicamente, possiamo comunque riassumere così l'operato della routine di lettura:

▲ Apre il file in input, e legge (con Input$) nel formato con cui erano stati salvati (2 byte per Larghezza ed Altezza, 1 byte per Profondità e Tipo), i primi 6 byte, li trasforma in valori interi (Con CVI ed ASC) e li assegna alle variabili già esaminate in fase di salvataggio.

▲ Apre uno schermo adoperando come parametri i valori appena ottenuti, quindi apre anche una finestra su di esso.

▲ Ricava gli indirizzi dei Bitplane dello schermo secondo le modalità illustrate nel riquadro.

▲ Legge i byte relativi alle componenti di colore, li divide per 100 (per ripristinare il loro valore originale), e li assegna alle relative Palette.

▲ Carica, 4 per volta, il contenuto dei Bitplane e li Poka (con Pokel) agli indirizzi prima ottenuti.

▲ Emette un Beep, ed attende la pressione di un tasto per concludere la visualizzazione della schermata.

Un (quasi) trucco: se non si vuole mostrare il formarsi dei vari Bitplane (lo stesso dicasi per il listato 4 che vedremo tra breve), basta aprire un altro schermo (magari con finestra vuota) inserendo un'istruzione Screen (con i parametri che si preferiscono) immediatamente prima della riga di commento "parametri colore", e chiuderlo *(Screen Close x)* dopo il Beep posto in fondo al listato.

TURBO!

Se non fosse per la velocità di esecuzione, quanto finora visto sarebbe già un ottimo risultato.

Ma, una volta capito il meccanismo, non è poi così difficile ovviare. Mantenendo lo stesso algoritmo, tutto ciò che occorre fare è modificare le modalità di accesso al file.

In altre parole, salvare e leggere i dati non più tramite i comandi del basic *(Open, Print# ed Input$)*, ma ricorrendo alle routine del *Dos*.

Il vantaggio consiste nel fatto che, adoperando le funzioni *xRead* e *xWrite* della *Dos Library* (si veda riquadro), si possono inviare (o prelevare) grosse quantità di dati tutte in una volta, e non byte per byte come viene fatto nei listati 1 e 2.

Per verificarne l'efficacia, si modifichino i due programmi già esaminati in modo che risultino uguali ai listati 3 e 4 (o si copino direttamente questi ultimi). Prima di mandarli in esecuzione, è necessario anche predisporre il file *Dos.bmap*, copiandolo dalla directory *BasicDemos* del dischetto *Extras* alla directory che sarà quella corrente al momento dello *Start* (di solito, la stessa ove è memorizzato il programma). In alternativa, lo stesso può essere posto all'interno della directory *Libs* del dischetto principale, ovvero quello con cui si è lanciato il sistema (di solito, Workbench).

Impartito il Run al listato 3 (per memorizzare) e, successivamente, al listato 4 (per visualizzare), la differenza di velocità risulterà palpabile.

Digitate con attenzione i vari listati e, soprattutto, consultate il manuale per ottenere maggiori informazioni

CERCASI MAPPA

Quando si intende manipolare una schermata, ed è proprio il caso nostro, risulta determinante conoscere l'indirizzo di inizio di ogni suo Bitplane.

La cosa non è difficile da attuarsi, ma... occorre aver fede, accettando ciò che in questa sede verrà esposto in modo non del tutto esaustivo (l'argomento è alquanto complesso: se ne riparlerà).

Come si è talvolta accennato, ad ogni schermo aperto corrisponde in memoria una raggruppamento di locazioni, definito *Struttura*, che contiene tanto dei dati (p. es. larghezza ed altezza dello schermo), quanto dei puntatori ad altre *Strutture*, che specificano ulteriormente altri aspetti della visualizzazione.

Senza troppo approfondire, basti sapere che, una volta noto l'indirizzo ove è memorizzata la Struttura *Screen* (quella dello schermo aperto), è possibile risalire anche all'indirizzo della Struttura *Bitmap*, che, a sua volta, fornirà la locazione di inizio di ogni singolo BitPlane.

Per sapere dove è locata la struttura inerente lo Schermo, si deve (da basic) aprire una finestra su di esso, e quindi, come operazione preliminare, leggere il valore fornito dalla funzione *Window (7)*.

Nel listato 1, p. es., la finestra viene aperta già nelle prime righe, per cui è poi sufficiente (nella sezione "bitplanes") sommare 46 a Window (7) per ottenere l'indirizzo della locazione ove è memorizzato l'indirizzo di inizio della Struttura schermo (se pensate di esservi persi, il labirinto è ancora a... Level 1).

In definitiva, questo benedetto indirizzo a quattro byte (Long) sarà leggibile con...

sch& = PEEKL (WINDOW (7) + 46)

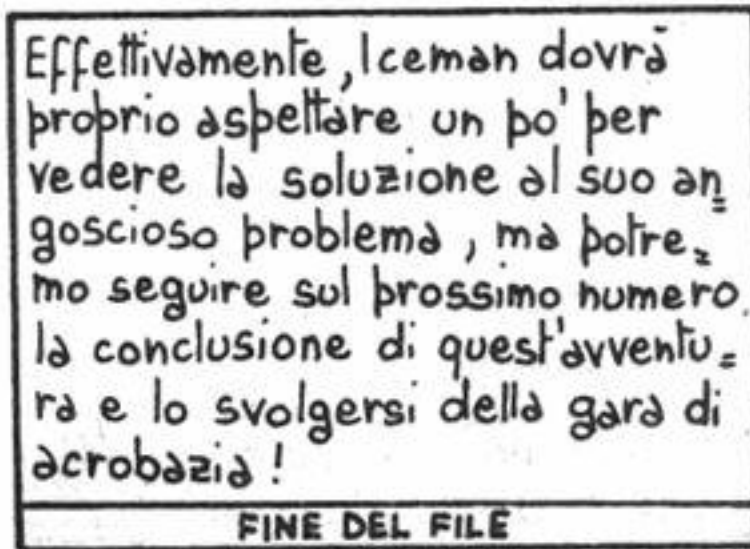
Ora, giocando un po' alle scatole cinesi (si seguano le operazioni nel listato 1), si può risalire ad un'altra Struttura, chiamata *RasterPort*, al cui interno è presente l'indirizzo dell'ennesima Struttura: quella che riguarda la *BitMap*.

Qui (era ora!) sono memorizzati, in forma *Long* (sempre 4 byte), a cominciare dal nono byte (il n. 8 contando anche lo zero), gli indirizzi di inizio di ogni BitPlane.

Con ciò, almeno parzialmente, è anche soddisfatta la curiosità di un lettore che, forse ancora intriso di cognizioni... ad 8 bit (che tenerezza!), ci ha chiesto l'ubicazione "dei puntatori all'area di schermo su Amiga".

Visto che roba?

Effettivamente, Iceman dovrà proprio aspettare un po' per vedere la soluzione al suo angoscioso problema, ma potremo seguire sul prossimo numero la conclusione di quest'avventura e lo svolgersi della gara di acrobazia!

FINE DEL FILE

L'articolo non è certo semplice a causa degli argomenti affrontati: leggetelo almeno un paio di volte

TUTTI IN LIBRERIA

Nei listati 3 e 4 di queste pagine, viene sfruttata la Libreria Dos per accedere, tanto in input che in output, ad un file.

In particolare, si adoperano le funzioni *xOpen, xRead, xWrite e xClose* (la "x" serve solo per distinguerli dagli omonimi comandi basic).

Per un loro corretto uso, è necessario anzitutto usare *Declare Function* come visibile nei listati, ad eccezione di *xClose*, che non lo richiede.

Poi, una volta aperta la libreria (istruzione Library "Dos.library"), si può cominciare con l'aprire il file che dovrà contenere la schermata.

Per farlo, va usato xOpen, secondo la sintassi...

canale& = xOpen (punt&, modo)

...dove *punt&* (puntatore) specifica l'indirizzo di una variabile stringa, con *Chr$(0)* come ultimo carattere, indicante il nome del file da aprire (in scrittura o lettura, non fa differenza).

Trattandosi di una stringa, l'indirizzo della variabile è ottenibile, in basic, tramite la funzione *Sadd* (si veda il manuale).

"Modo", invece, può assumere due valori: 1005 e 1006.

Con 1005, si apre un file già esistente, mentre con 1006 viene aperto un nuovo file: se ne esiste già uno con lo stesso nome, il suo precedente contenuto verrà totalmente cancellato.

La variabile *canale&*, nel caso si sia verificato un errore, varrà 0 oppure -1, altrimenti conterrà un particolare valore da inserire (come vedremo) in tutte le operazioni che riguardano il file.

Per l'input e l'output vanno adoperati, con identiche modalità, *xRead* (lettura) e *xWrite* (scrittura) con la seguente sintassi:

Lettura& = xRead& (canale&, buffer&, numerobyte&)

Scrittura& = xWrite (canale&, buffer&, numerobyte&)

Le variabili Long (a 4 byte) Lettura e Scrittura, anche qui indicheranno un errore (nei listati non valutato) se uguali a 0 oppure -1, altrimenti conterranno il numero di bytes letti.

"Canale&", altri non è che il valore ottenuto in fase di apertura del file, mentre *"numerobyte&"* indica proprio ciò che pensate: il numero di byte che devono essere letti o scritti.

"Buffer&" deve contenere l'indirizzo di inizio di un'area di memoria dalla quale verrà prelevato (per *xWrite*) o nel quale verrà inserito (per *xRead)* il numero di dati di cui sopra.

Questo indirizzo, nel caso dei listati in esame, viene fatto coincidere prima con quello di una variabile stringa fittizia (riempita di 100 spazi con *Space$*); una tecnica non proprio ortodossa, ma evita di adoperare altre librerie per riservare ai nostri scopi una zona di memoria: in fondo, una variabile occupa anch'essa un certo numero di locazioni (leggi: è un buffer), e l'interprete del basic evita la fatica di renderle inviolabili (ci pensa lui).

Nella sezione *"crea buffer"* dei listati, l'indirizzo di *Buffer&* viene anche forzato ad un numero pari, per consentire un successivo corretto uso dei comandi *PeekW* e *PokeW* (ennesima sfogliatina al manuale).

Quando, invece, si dovranno leggere (o "spedire") i Bitplane, è proprio il loro l'indirizzo ad essere segnalato come Buffer: in tal modo,

i dati verranno manipolati con estrema velocità, grazie anche alla possibilità di fornire alla funzione di Libreria (come parametro numerobyte&) l'intera dimensione di ogni Bitplane.

Con *Call xClose&(canale&)*, ogni commento è superfluo, il file specificato da *Canale&* viene chiuso.

Librerie a parte, valgono le stesse considerazioni già fatte in precedenza, anche nei confronti di eventuali modifiche da apportare ai programmi.

Sull'uso delle Librerie, ci sarebbe ancora molto da dire, ma... troppa carne al fuoco rischia di confondere le idee già assimilate.

D'altra parte, una volta descritte le basi che riguardano la manipolazione degli schermi, sarà per noi inevitabile una ulteriore trattazione (lasciando più spazio alle Librerie), per vedere come rendere la grafica così generata compatibile con i formati standard presenti sul mercato (l' IFF dei vari *Deluxe Paint, GraphiCraft*, ecc.).

Appuntamento, dunque, ad un prossimo numero, ma senza dimenticare che le tecniche esaminate in questa sede, per quanto agiscano elettivamente in ambito basic, risultano, alla resa dei conti, più efficienti e veloci del trattamento (sempre in Basic) di schermate organizzate nello standard *Iff/Ilbm*..

La grafica di Amiga, dopo alcune difficoltà iniziali, è in grado di dare molte soddisfazioni ad i suoi utenti

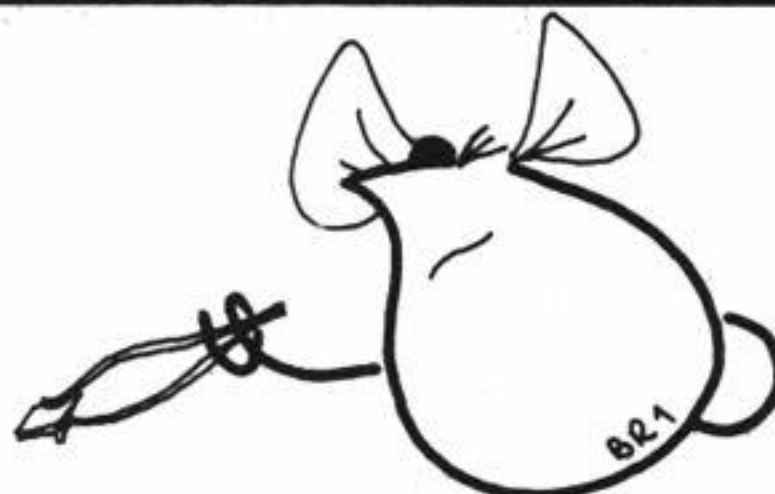
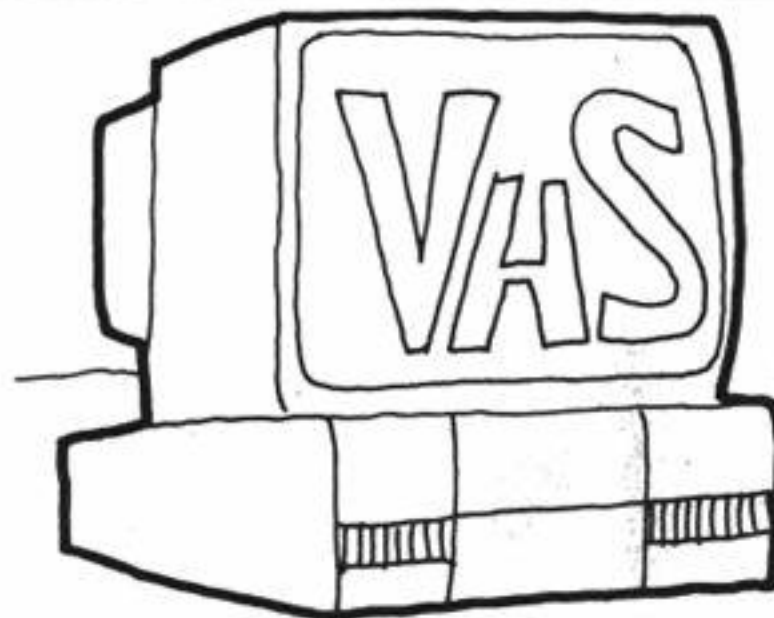
```
'              L I S T A T O   1
'-----------------------------------------------
'       Genera una schermata grafica e la
'       memorizza  su  periferica,  senza
'       sfruttare librerie di sistema.
'===============================================

DATA 0,0,0      :REM palette 0 = nero
DATA 0,.93,.87  :REM palette 1 = acqua
DATA .93,.2,0   :REM palette 2 = rosso
DATA 1,.13,.93  :REM palette 3 = viola
largh=320:alt=256:prof=2:tipo=1
colori=2^prof:DIM r(colori),g(colori),b(colori)
SCREEN 1,largh,alt,prof,tipo:WINDOW 2,,,0,1
FOR x=0 TO colori-1
   READ r(x),g(x),b(x):PALETTE x,r(x),g(x),b(x)
NEXT x

'=============== SCHERMATA DEMO =================
FOR x=1 TO 100 STEP 8:CIRCLE (160,125),x,2:NEXT
COLOR 3,0 : DIM pat%(1)
pat%(0)= &H0:pat%(1)= &HFFFF:PATTERN &HFFFF,pat%
AREA (10,210):AREA STEP (20,0):AREA STEP (0,-190)
AREA STEP (-20,0):AREAFILL
pat%(0)=&HAAAA:pat%(1)=&HAAAA:PATTERN &HFFFF,pat%
AREA (50,240):AREA STEP (230,0):AREA STEP (0,-20)
AREA STEP (-230,0):AREAFILL

'=============== SALVA SCHERMATA ================
OPEN "pic" FOR OUTPUT AS 1
'-------------- parametri schermo --------------
PRINT #1,MKI$(largh);MKI$(alt);
PRINT #1,CHR$(prof);CHR$(tipo);
'-------------- parametri colore ---------------
FOR x=0 TO colori-1
  PRINT #1,CHR$(r(x)*100);CHR$(g(x)*100);
  PRINT #1,CHR$(b(x)*100);
NEXT x
'--------------- bitplanes ---------------------

sch&=PEEKL(WINDOW(7)+46):rastport&=sch&+84
bitmap&=PEEKL(rastport&+4)
FOR x=0 TO prof-1
  bitpl&(x) = PEEKL(bitmap&+8+(x*4))
NEXT
dimensione=(largh/8)*alt
FOR x=0 TO prof-1
  FOR y=0 TO dimensione-1 STEP 4
    PRINT #1, MKL$(PEEKL(bitpl&(x)+y));
  NEXT y
NEXT x

'===============================================
fine:
BEEP:CLOSE:WINDOW CLOSE 2:SCREEN CLOSE 1:END
```

```
'              L I S T A T O   2
'-----------------------------------------------
' Carica e visualizza una schermata salvata
' con il listato 1 oppure con il listato 3
' senza ricorrere a librerie di sistema
'===============================================
OPEN "pic" FOR INPUT AS 1
'============ parametri schermo ================
largh=CVI(INPUT$(2,1)):alt=CVI(INPUT$(2,1))
prof=ASC(INPUT$(1,1)):tipo=ASC(INPUT$(1,1))
SCREEN 1,largh,alt,prof,tipo:WINDOW 2,,,0,1
sch&=PEEKL(WINDOW(7)+46):rastport&=sch&+84
bitmap&=PEEKL(rastport&+4)
FOR x=0 TO prof-1
  bitpl&(x)=PEEKL(bitmap&+8+(x*4))
NEXT
'=========== parametri colore ==================
colori=2^prof
FOR x=0 TO colori-1
  r=ASC(INPUT$(1,1))
  g=ASC(INPUT$(1,1))
  b=ASC(INPUT$(1,1))
  PALETTE x,r/100,g/100,b/100
NEXT
'============== bitplanes ======================
dimensione=(largh/8)*alt
FOR x=0 TO prof-1
  FOR y=0 TO dimensione-1 STEP 4
    POKEL bitpl&(x)+y, CVL(INPUT$(4,1))
  NEXT y
NEXT x : BEEP
'===============================================
loop:
a$=INKEY$:IF a$="" THEN GOTO loop
CLOSE:WINDOW CLOSE 2:SCREEN CLOSE 1:END
```

```
'                 L I S T A T O   3
'-----------------------------------------------
' Memorizzazione veloce di una schermata grafica
'===============================================
DATA 0,0,0       :REM palette 0 = nero
DATA 0,.93,.87   :REM palette 1 = acqua
DATA .93,.2,0    :REM palette 2 = rosso
DATA 1,.13,.93   :REM palette 3 = viola
largh%=320:alt%=256:prof=2:tipo=1
'-----------------------------------------------
DECLARE FUNCTION xOpen& LIBRARY
DECLARE FUNCTION xWrite& LIBRARY
LIBRARY "dos.library"
colori=2^prof:DIM r(colori),g(colori),b(colori)
SCREEN 1,largh%,alt%,prof,tipo:WINDOW 2,,,0,1
FOR x=0 TO colori-1
  READ r(x),g(x),b(x):PALETTE x,r(x),g(x),b(x)
NEXT x
'=============== SCHERMATA DEMO =================
FOR x=1 TO 100 STEP 8:CIRCLE (160,125),x,2:NEXT
COLOR 3,0 : DIM pat%(1)
pat%(0)= &H0:pat%(1)= &HFFFF:PATTERN &HFFFF,pat%
AREA (10,210):AREA STEP (20,0):AREA STEP (0,-190)
AREA STEP (-20,0):AREAFILL
pat%(0)=&HAAAA:pat%(1)=&HAAAA:PATTERN &HFFFF,pat%
AREA (50,240):AREA STEP (230,0):AREA STEP (0,-20)
AREA STEP (-230,0):AREAFILL
'================ apertura file =================
nome$="pic"+CHR$(0)
canale&=xOpen&(SADD(nome$),1006)
IF canale&<=0 THEN
  WINDOW OUTPUT 1:PRINT"FILE ERROR":GOTO fine
END IF
'---------------- crea buffer ------------------
buffer$=SPACE$(100):buffer&=SADD(buffer$)
IF (buffer&/2) <> INT(buffer&/2) THEN
  buffer&=buffer&+1
END IF
'--------------- parametri schermo --------------
POKEW buffer&,largh%:POKEW buffer&+2,alt%
POKE buffer&+4,prof:POKE buffer&+5,tipo
scrive& = xWrite&(canale&,buffer&,6)
'--------------- parametri colore ---------------
colori=2^prof:totrgb&=colori*3
FOR x=0 TO colori-1
  POKE (buffer&+(x*3)),r(x)*100
  POKE (buffer&+(x*3)+1),g(x)*100
  POKE (buffer&+(x*3)+2),b(x)*100
NEXT
scrive& = xWrite&(canale&,buffer&,totrgb&)
'------------------ bitplanes -------------------
sch&=PEEKL(WINDOW(7)+46):rastport&=sch&+84
bitmap&=PEEKL(rastport&+4)
FOR x=0 TO prof-1
  bitpl&(x)=PEEKL(bitmap&+8+(x*4))
NEXT
dimensione&=(largh%/8)*alt%
FOR x=0 TO prof-1
  scrive&=xWrite&(canale&,bitpl&(x),dimensione&)
NEXT x : BEEP
'===============================================
fine:
CALL xClose&(canale&):LIBRARY CLOSE
WINDOW CLOSE 2:SCREEN CLOSE 1:END
```

```
'                 L I S T A T O   4
'-----------------------------------------------
' Caricamento superveloce di una schermata
'salvata con il listato 1 o con il listato 3
'===============================================
DECLARE FUNCTION xOpen& LIBRARY
DECLARE FUNCTION xRead& LIBRARY
LIBRARY "dos.library"
'============= apertura file ===================
nome$="pic"+CHR$(0)
canale&=xOpen&(SADD(nome$),1005)
IF canale&<=0 THEN
  PRINT "NON POSSO APRIRE IL FILE!":GOTO fine
END IF
'------------  crea  buffer  -------------------
buffer$=SPACE$(100):buffer&=SADD(buffer$)
IF (buffer&/2)<>INT(buffer&/2) THEN
  buffer&=buffer&+1
END IF
'============== parametri schermo ==============
lettura& = xRead&(canale&,buffer&,6)
largh=PEEKW(buffer&):alt=PEEKW(buffer&+2)
prof=PEEK(buffer&+4):tipo=PEEK(buffer&+5)
SCREEN 1,largh,alt,prof,tipo:WINDOW 2,,,0,1
sch&=PEEKL(WINDOW(7)+46):rastport&=sch&+84
bitmap&=PEEKL(rastport&+4)
FOR x=0 TO prof-1
  bitpl&(x)=PEEKL(bitmap&+8+(x*4))
NEXT
'============ parametri colore =================
colori=2^prof:totrgb&=colori*3
lettura& = xRead&(canale&,buffer&,totrgb&)
FOR x=0 TO colori-1
  r=PEEK(buffer&+(x*3))
  g=PEEK(buffer&+(x*3)+1)
  b=PEEK(buffer&+(x*3)+2)
  PALETTE x,r/100,g/100,b/100
NEXT
'================== bitplanes ==================
dimensione&=(largh/8)*alt
FOR x=0 TO prof-1
  lettura&=xRead&(canale&,bitpl&(x),dimensione&)
NEXT x : BEEP
'=============== fine programma ================
loop:
A$=INKEY$:IF A$="" THEN GOTO loop
fine:
CALL xClose&(canale&):LIBRARY CLOSE
WINDOW CLOSE 2:SCREEN CLOSE 1:END
```

GRATUITO.
VANTAGGIOSO.
INTERESSANTE.

RARAMENTE CAPITA DI POTER ASSOCIARE QUESTE QUALITA' AD UN SERVIZIO.

PERSONAL COMPUTER ANNUNCIA UN'INIZIATIVA CON TUTTE QUESTE CARATTERISTICHE.

DAL 2 APRILE SARA' OPERATIVA BBSYSTEMS, LA NUOVA BANCA DATI PROMOSSA DA SYSTEMS EDITORIALE.

UN SERVIZIO GRATUITO, VANTAGGIOSO, INTERESSANTE.

SULLO SCAFFALE

Diamo uno sguardo ai libri-guida degli utenti di amiga

L. Callegari, M. Feletto

Linguaggio C per Amiga

GR Edizioni, Via XXV Aprile, 3
20097 S.Donato MI
Lire 60.000 (con disco)

Il C rappresenta certamente il linguaggio di elezione per chiunque desidera lavorare professionalmente con Amiga: presenza di ottimi pacchetti commerciali *(Lattice, Aztec)*, ampia documentazione sia su carta sia in software di pubblico dominio, grande velocità di elaborazione. Il volume è nato quindi per tentare di colmare una evidente lacuna editoriale: nessun testo è stato sinora pubblicato in Italia per insegnare specificamente il **C** a chi possiede Amiga: o si parla del C in ambienti più o meno esotici (UNIX, MS/DOS), o si parla del C per Amiga, trascurando chi non conosce le basi di questo linguaggio relativamente moderno.

Da quanto si legge anche nella prefazione, il libro è concepito come un testo che, partendo dall'assunzione che il lettore abbia solo una infarinatura di programmazione con qualche altro linguaggio di base (Basic, ad esempio), e trattando quindi le basi fondamentali della programmazione in C (senza comunque mai dimenticare lo standard ANSI proposto e le implementazioni effettive presenti per Amiga), arriva sino alla descrizione delle funzioni specifiche delle librerie di Amiga, aggiornate alla V1.3 del sistema operativo, riportando addirittura i "bug" più noti.

Gli autori, due giovani che lavorano nel mondo della microinformatica da più di sei anni (*Luigi Callegari* è anche collaboratore delle riviste Systems da altrettanto tempo), consentono di farvi evitare l'acquisto di libri stranieri (ROM Kernel Manual) che spiegano soltanto come si inizia a programmare Amiga.

Chi è già esperto dovrebbe apprezzare un unico testo, *in italiano*, contenente la descrizione dettagliata delle librerie di Amiga.

Il libro è suddiviso in sette capitoli (Il linguaggio C, Exec, Grafica, Testo e Fonti, Layers, Intuition, AmigaDOS) con alcune appendici, tra le quali gli elenchi delle opzioni dei compilatori C commerciali (Lattice e Aztec) ed un utile glossario informatico per utenti Amiga.

Dopo il lungo e dettagliato capitolo introduttivo, che spiega i concetti del linguaggio C partendo da zero, si passa ai capitoli sull'uso specifico con Amiga.

Si spiegano dapprima genericamente i compiti svolti dalle funzioni di una libreria, con eventuali collegamenti con l'hardware e connessi risvolti teorici, poi si riportano alcuni esempi d'uso pratico in programmi didattici con funzioni generiche (spesso riciclabili in altri programmi) ed infine viene riportata la descrizione sistematica, in ordine alfabetico, di tutte le funzioni, con tanto di formati C dei parametri di entrata ed uscita.

Il libro contiene parecchi listati di programmi contenenti funzioni generiche e riciclabili, inseriti nel dischetto accluso, commentati diffusamente ed adattati ad essere compilati con i pacchetti più moderni senza alcuna modifica: Lattice C V5 e Aztec C V3.6.

Un altro innegabile pregio del libro sta nell'essere il primo *scritto da italiani per italiani*, cioè usando la nostra filosofia ed uno stile di scrittura ed esposizione non tradotto, una volta tanto, dall'inglese o dal tedesco, senza quindi l'immancabile inserimento di aree di confusione da parte del traduttore più o meno esperto.

Dulcis in fundo, a dimostrazione di quanto sia vero che Amiga non è solo una macchina per videogiochi, ma uno strumento di lavoro professionale, gli autori hanno prodotto il libro interamente con Amiga, avvalendosi dei pacchetti Professional Page e Professional Draw della Gold Disk per impaginarlo e produrre le pellicole per la stampa tipografica. Il testo è adatto alle fasce, di utenti Amiga, comprese tra due estremi: i principianti (che vogliono programmare in C partendo da zero), e gli esperti (per avere a portata di mano un pratico ed esauriente manuale di riferimento).

Commodore Amiga Inc.

Amiga Hardware Reference Manual Revised and Updated

Addison Wesley Publishing
Lire 60.000 (circa)

La serie di libri tecnici prodotti dalla stessa Commodore per l'Amiga si è arricchito di questa nuova edizione del manuale, famoso sia per le preziose informazioni contenute, sia per le evidenti imprecisioni e carenze presenti. Il testo originale fu scritto per la V1.1 da quattro autori; la nuova edizione è stata riscritta, partendo da quel testo, con la revisione di sette bravi programmatori, alcuni dei quali "esterni" alla Commodore stessa.

Il volume fornisce informazioni sulle capacità grafiche, sonore e di interfacciamento di Amiga da un punto di vista strettamente tecnico. In particolare si rivolge a tutti coloro i quali vogliono avere le basi cognitive per programmare direttamente in assembly a "basso livello", senza usare le routines descritte nei ROM Kernel Manual. Inoltre è chiaramente rivolto anche a chi vuole progettare interfacce o circuiti elettronici da collegare ad Amiga, oltre ovviamente a tutti i curiosi di sapere come funziona a livello hardware il proprio computer.

Il libro conta 385 pagine, suddivise in otto capitoli, alcune appendici ed un glossario.

Il primo capitolo introduce gli schemi hardware di base di Amiga, il secondo invece spiega dettagliatamente il funzionamento del coprocessore *Copper* e co-

me lo si programma per gestire grafica e sonoro. Il terzo capitolo illustra come si creano, si visualizzano e si manipolano i *playfields* e come si possono ottenere visualizzazioni a colori ed in varie risoluzioni e modi, mentre il quarto capitolo tratta gli otto sprite hardware gestiti in DMA, spiegando anche come si programmano le apposite strutture di dati previste dal sistema. Il quinto capitolo è dedicato all'audio: suoni semplici, campionati, stereofonici, modulati e più o meno complessi. Il sesto capitolo tratta il Blitter, evidenziando come sia possibile sfruttarlo per ottenere animazioni e tracciature grafiche velocissime.

Il settimo capitolo spiega come si devono usare i registri hardware mappati in memoria per controllare le interruzioni, le collisioni tra sprites, la gestione del DMA e le varie parti hardware di Amiga. L'ottavo capitolo, infine, illustra come si può fare colloquiare l'hardware di Amiga con il mondo esterno attraverso la porta seriale, parallela, mouse, joystick, porta di espansione e uscita video, riportando anche utili cenni sul funzionamento del controller di dischi e sullo slot di espansione della RAM.

Le appendici riportano i vari registri mappati in memoria, con indirizzi e significati, mappe di memoria, descrizioni dei connettori interni ed esterni, specifiche sulla tastiera, sulla CIA, il clock ed altro ancora. Si tratta insomma di un libro consigliabile a chiunque abbia già un buon patrimonio di conoscenza su Amiga e vuole approfondirla ulteriormente, purchè abbia una mimima competenza su microelettronica per elaboratori e sappia bene l'inglese informatico.

Commodore Amiga Inc.

Amiga ROM Kernel Manual, Includes and Autodocs

Addison Wesley Publishing
Lire 70.000 (circa)

Questo libro non sostituisce i due classici tomi *Amiga ROM Kernel Manual*, ma fornisce una quantità di informazioni supplementari aggiornate alla V 1.3 del sistema operativo. Contiene sommari di tutte le funzioni delle librerie di Amiga (escluse quelle di AmigaDOS), i listati di tutti i files di inclusione forniti con i compilatori **C** e gli assemblatori, gli elenchi di tutti i nomi di strutture di dati riservate, informazioni sullo standard IFF e molto altro ancora.

Le oltre *settecento* pagine di cui è composto sono infatti suddivise in ben undici capitoli.

Il primo serve da introduzione, illustrando dettagliatamente alcuni aspetti fondamentali della programmazione pratica in C od Assembler di Amiga, evidenziando le restrizioni da osservare per garantire che il proprio codice rimanga compatibile con future versioni del sistema operativo. Il secondo capitolo elenca tutte le funzioni di libreria, mentre il terzo spiega la gestione dei device via software ed il quarto in modo analogo come si programmano le *risorse*. Il quinto capitolo elenca tutti i files di inclusione dei compilatori C, così come il sesto riporta gli stessi files per assemblatori. Il settimo illustra le funzioni contenute nelle librerie di *liking*, con alcuni semplici esempi di utilizzo.

L'ottavo capitolo riporta listati esemplificativi di libreria e device autocostruiti, mentre il nono capitolo è una guida di riferimento di tutte le strutture di dati e funzioni usate in C. Il decimo è la copia del testo fatto circolare dalla Electronic Arts per spiegare lo standard IFF, con un completo listato esemplificativo in C e la spiegazione delle funzioni previste da librerie di pubblico dominio. L'undicesimo capitolo è l'elenco sistematico delle funzioni.

Come detto prima, il volume è da affiancare ai ROM Kernel Manual originali ed è da considerarsi un testo praticamente irrinunziabile, soprattutto come riferimento durante la stesura del software (mentre i ROM Kernel Manual originali servivano anche come testi propedeutici) per chi vuole scrivere programmi C od Assembler con Amiga.

Bleek, Maelger, Waltner

Amiga Tricks and Tips

FTE Editions, Via Sassoferrato, 1.
20135 Milano
Lire 45.000

Il volume, di circa 480 pagine, è colmo di trucchi e suggerimenti utili per impratichirsi nell'uso di Amiga. Inizialmente tratta tutti i comandi della versione 1.3 del CLI, poi passa ad alcune preziosissime informazioni sull'uso del sistema FFS e dei nuovi *device* ed *handler* previsti dall'ultima versione del sistema operativo, non trattati dalla bibliografia ufficiale Commodore.

Il pezzo forte sono comunque le dozzine di listati in Amigabasic che illustrano come gestire gadget, finestre, grafica bi/tridimensionale, icone, modi grafici speciali (*Extra Habrite* e *Ham*), la gestione dello standard IFF, come calcolare i benchmark dei propri programmi ed ottimizzarli, come eseguire degli *Input* multitasking controllati, come si usano le funzioni delle librerie assembly di Amiga, e tanto altro ancora, il tutto sempre rigorosamente da *AmigaBasic*.

In coda seguono numerose informazioni *sparse*, veramente da apprezzare una per una, talvolta già note ma spesso del tutto innovative e comunque comodamente ed ordinatamente elencate: come ottenere CLI colorati e con vari stili, come trovare le frasi nascoste nel Kickstart e nel Workbench, come ottenere (ed evitare) strane *Guru Meditation*, particolarità dell'uso di Preferences, combinazioni di ESCape per il CLI e via spigolando.

Non mancano alcune note sull'interfacciamento di routines scritte in vari linguaggi, Assembler o C, con Amigabasic, sempre completate da ottimi listati commentati ampiamente.

Questo libro è un "best seller" della *Abacus / Data Becker* in Inghilterra e Germania, paesi notoriamente evoluti informaticamente, e già questo sarebbe un'ottima garanzia per l'acquisto. Il prezzo sembra piuttosto contenuto, considerati gli standard di mercato e la quantità (e qualità) delle informazioni riportate.

La traduzione è in qualche punto stilisticamente imperfetta (forse perchè l'edizione italiana è nata dall'assemblaggio delle edizioni tedesche ed inglese del libro) ma sempre tecnicamente precisa, evidentemente svolta da una persona che conosce molto bene Amiga.

Facendo il raffronto con le edizioni originali, si nota che molti temi sono stati ampliati facendo ricorso ad informazioni fatte circolare dalla Commodore dopo alcuni mesi dalla loro pubblicazione, cioè qualche mese prima della edizione nostrana, particolare apprezzabile in Italia dove, normalmente, molti libri escono anche ad anni di distanza senza alcun aggiornamento svolto dai loro curatori.

INSEGNIAMO AI NOSTRI COMPUTER A PARLARE L'ASCII

Un programma certamente utile per chi già dispone del C/64 Emulator o ha bisogno di un "copiatore" personalizzato

di Valentino Spataro

Generalmente, quando usiamo un programma che gira sul C/64, non pensiamo che i files prodotti non saranno leggibili da Amiga. Oggi, però, grazie ad Amiga, al C/64 Emulator II ed al programma qui proposto, è possibile trasferire automaticamente tutti i vostri archivi e testi da C/64 ad Amiga e viceversa.

LO STANDARD ASCII

L'ASCII è uno standard che l'IBM impose quando ancora era talmente forte da dettare legge. Da quei tempi tutti i personal registrano testi in tale formato. Amiga non è rimasta insensibile al fenomeno adottando lo standard, almeno per quanto riguarda le lettere dell'alfabeto ed altri caratteri tra i più diffusi. Il glorioso C/64, più antico, ha un suo Ascii particolare.

Oggi, teoricamente, è facile gestire testi, prodotti con un certo programma, mediante un altro programma, che utilizza un formato che, però, non risulta compatibile con il testo originale.

Pertanto assistiamo ad una situazione contraddittoria: se da una parte si sviluppano programmi in grado di emulare altri computer (e quindi di utilizzare i programmi e i dati che altrimenti occorrerebbe convertire o ridigitare) d'altro canto non si sviluppa una mentalità, tra i programmatori, volta ad utilizzare formati compatibili. Ciò si verifica soprattutto nel campo dei database, dove al dominio del *DB III* corrisponde una totale anarchia per quanto riguarda i programmi meno diffusi.

Per Amiga bisogna ammettere che il formato IFF, sia in campo sonoro che grafico, è uno standard di fatto tra la maggior parte dei programmi. Convertitori di dati da un formato ad un altro sono comunque generalmente inclusi nei package che usano standard diversi.

Oggi la mancanza di uno standard effettivamente impostosi a larga scala nel campo dei database, per quanto riguarda il mercato Amiga, è dannoso. Pensiamo infatti a programmi come *Cross Dos* che permette ad Amiga di riconoscere e leggere dischi da 3.5 da

DA AMIGA A 64. E VICEVERSA

Chi si avvicina ad Amiga, dopo esser "passato" per il C/64, avverte spesso l'esigenza di non abbandonare del tutto il vecchio computer, sia per motivi affettivi sia per vere e proprie necessità.

Di solito, infatti, sono moltissimi i files generati con il C/64 che, di varia natura (sprattutto testi eseguiti con un word processor) possono esser recuperati in ambiente Amiga e trattati adeguatamente.

Perchè, ad esempio, non sfruttare subito le straordinarie possibilità di impaginazione offerte da un vero *Desk Top Publishing* per Amiga?

Potrebbe capitarci, infatti, di re-impaginare un testo che presentava alcuni limiti di formattazione con il word processor che usavamo di solito con il C/64.

Ma ecco, quindi, che il desiderio di riutilizzare vecchi files (e, perchè no, anche semplici programmi basic interpretabili da Abasic) si scontra clamorosamente non solo con il differente formato del supporto magnetico ma, addirittura, con lo stesso standard di memorizzazione dei caratteri alfanumerici.

Ben vengano, quindi, i preziosi *tools* di conversione, di adattamento, di modifica o di semplice "interpretazione" di files già in nostro possesso.

E non si può concludere in altro modo se non con un invito, rivolto ai nostri lettori, per verificare se i nostri problemi sono anche i vostri:

Avete realizzato particolari routine che consentono di "trattare" files generati da altri programmi? Avete, nel cassetto, un convertitore di files Easy Script C/64 in formato Wordstar Ms-Dos (da inviare, ovviamente, via Rs-232)?

Se pensate di essere utili ai nostri lettori, fatecelo sapere: telefonateci in Redazione, o meglio, servitevi della nostra BBS; è fatta apposta per avere un colloquio diretto con gli utenti Commodore.

720 Kb allo stesso modo dei dischi da 800 Kb di Amiga.

E' possibile quindi modificare direttamente su disco, in formato Ms Dos, con Notepad un testo creato creato con Wordstar. Però, per miopia dei programmatori, è necessario ridigitare intere liste di dati.

Se si sviluppasse una mentalità di questo tipo l'Amiga potrebbe proporsi addirittura come valida alternativa al mondo dei personal, per quanto riguarda almeno il mercato casalingo.

Ma non è detto che tutto questo non succeda, anzi. In attesa che i programmatori modifichino il loro modo di programmare per produrre prodotti più appetibili, proponiamo una routine LM per C/64, ma studiata per essere utilizzata anche dal C/64 Emulator.

Permette di tradurre, alla velocità del linguaggio macchina, i vostri file dal formato ASCII Amiga a quello del C/64, e viceversa.

IL PROGRAMMA

La routine in linguaggio macchina carica in memoria un file di qualsiasi tipo *(Seq, Usr, Rel, Prg)*. Il file può essere caricato sia in ASCII C/64 che in ASCII Amiga: è possibile inoltre convertirlo nel formato che ci interessa o anche solamente copiarlo così com'è.

La sintassi della routine è la seguente:

Sys Xxxx, So$, So, De, De$, Mode

dove *So$* e *De$* sono rispettivamente il nome del file sorgente e del file destinazione, *So* e *De* i valori dei device sorgente e destinazione (es. di valori: 4 per stampante, 3 per lo schermo, 8 per un drive, ecc.), *Mode* è un flag che indica se la conversione debba avvenire da C/64 ad Amiga (*Mode* <> 0 e minore di 256) o viceversa (*Mode* = 0).

La routine lm, il cui disassemblato è talmente elementare da non giustificare l'impiego di pagine preziose, è divisa in queste parti:

▲ prende i parametri passati con la SYS e li memorizza in RAM;

▲ carica il file prendendolo byte dopo byte e lo memorizza dalla locazione 10000 alla 40960. Possono essere caricati file per la lunghezza massima di 100 blocchi circa;

▲ prende i bytes e li spedisce sul device di output. Riportiamo qui di seguito la parte relativa in quanto tornerà utile per successive considerazioni:

```
LDA ($FD), Y
JSR CONVERTE
JSR CHROUT
```

tenendo presente che la prima istruzione carica nell'accumulatore il byte da spedire, la seconda converte il codice ASCII contenuto nell'accumulatore e la terza lo invia sul canale di output.

La routine è *quasi* interamente rilocabile: questo significa che solo usando il caricatore proposto nel demo potrete allocarla dove volete (naturalmente si sconsiglia l'area di memoria da 10000 a 40960 usata dalla routine); infatti esso provvede automaticamente a spostare la locazione a cui punta il *Jsr Converte* se la routine non viene allocata da 49152. Inoltre, essendo i bytes della routine oltre 350, si è preferito aumentare il numero dei controlli per facilitare l'individuazione di errori. Ricordiamo in proposito di registrare sempre il programma appena copiato prima di dare il Run.

Abbiamo detto anche che la routine può trasformarsi in un copiatore: studiando il listato troverete con facilità il punto in questione; qui spiegheremo la tecnica usata.

Premesso che il byte spedito sul canale aperto all'output è contenuto nell'accumulatore, e che la routine *Converte* (che potrete trovare nel disassemblato per l'assemblatore *Merlin* su directory di questo numero) provvede a modificare il contenuto dell'accumulatore: basta non richiamare questa routine.

Questo è stato ottenuto cambiando il codice dell'istruzione *Jsr Converte* (#32) con il codice dell'istruzione *Bit* (#44) che fa saltare l'esecuzione del programma due byte più avanti. E' come se si fossero cambiati i tre bytes dell'istruzione *Jsr Converte* con tre *Nop*. In questo modo il byte caricato dal device di input è lo stesso che viene spedito sul device di output. Per riabilitare la routine alla conversione basterà rimettere il codice #32

al suo posto (cioè alla locazione SA + 193 dove SA è l'inizio della routine: SA + 193, X dove X = 44 copia, X = 32 converte).

COME USARE IL DEMO

Il demo prevede l'utilizzo completo della routine, quindi basterà copiarlo, registrarlo, studiarlo e lanciarlo. Dovrete indicare i nomi dei files (se necessario: se l'output è su stampante e non su file non è importante il nome del file destinazione) ed i relativi device; se operare una conversione o una copia, e nel primo caso il tipo di conversione su cui operare. Considerando tutti i casi possibili potete quindi usare il programma per: 1) copiare, 2) stampare oppure 3) visualizzare su schermo i files A) così come sono o B) convertiti.

Il programma può essere lanciato sia da C/64 che da Amiga con il C/64 Emulator; in questo secondo caso la scelta del linguaggio macchina è stata obbligata per limitare i tempi di attesa.

Se usate il C/64 Emulator ricordiamo di non effettuare i collegamenti con le periferiche quando queste sono accese.

Notiamo qui di seguito che tale programma è composto da una interfaccia hardware (un semplice cavo con una presa parallela e un bus seriale alle estremità) la cui bontà non è riconosciuta universalmente. Personalmente non ho avuto inconvenienti di sorta, nè danneggiamenti, anche se studiando i vari tipi di cavi che mi sono passati per le mani, e sono la maggior parte di quelli in commercio, non sono riuscito a trovarne uno uguale all'altro; e questo a parità di funzionamento.

Quando collegate l'Amiga alle periferiche del C/64, scollegate prima quest'ultimo dalle stesse; con alcuni programmi può rivelarsi necessario spegnere la stampante, e comunque non parlo dei vari turbo e copiatori che, di solito, non funzionano; al contrario funziona perfettamente il programma *Transfer* presente sul disco del C/64 Emulator per trasferire files, anche se lento.

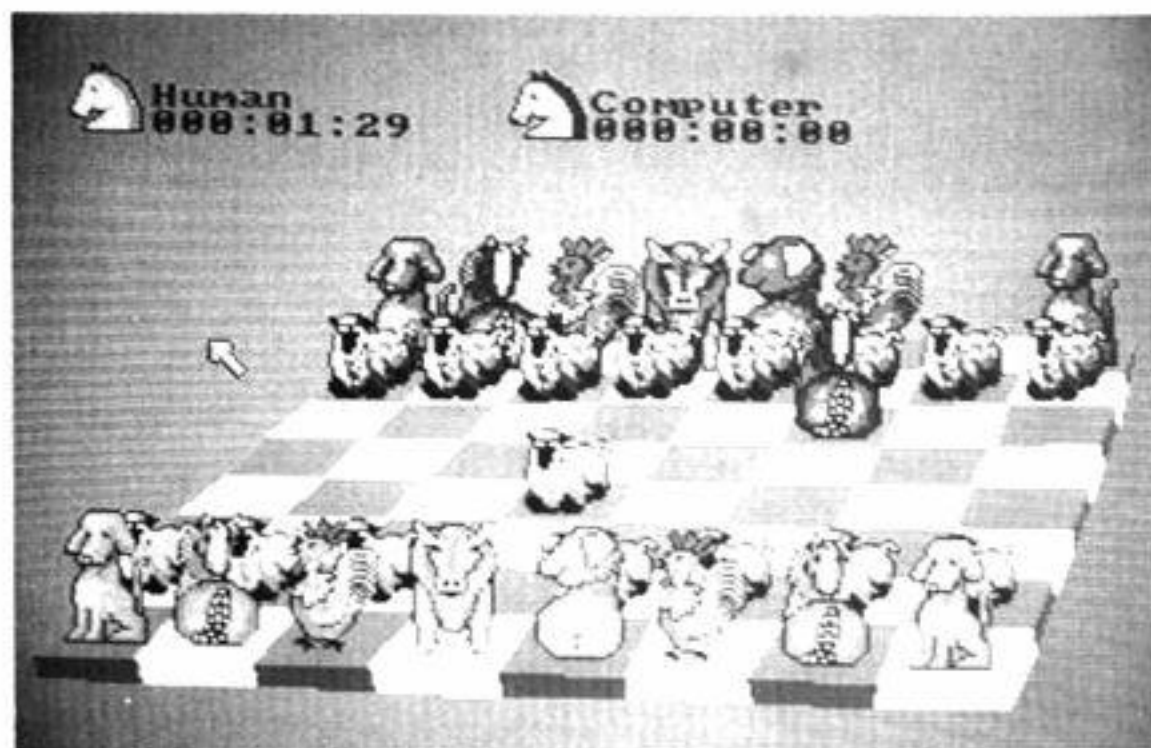
Riportiamo qui, per dovere di cronaca, che si sono presentati alcuni malfunzionamenti: talvolta, quando si usano i drive Amiga, non si riesce più a utilizzare le periferiche del C/64, in particolar modo la stampante. In tal caso si consiglia di passare il file convertito prima da disco 3.5 a disco 5.25 per poi stamparlo da C/64. Per chi dispone, invece, del programma e interfaccia *Amiprint* o simili, che dovrebbe provvedere ad utilizzare la Mps-803 con l'Amiga (sia in modo testo che in modo grafico) avvertiamo che alcuni esemplari di Mps-803 non funzionano, nè in modo testo, nè in modo grafico, pur essendo originali Commodore.

E' bene quindi, per evitare sorprese, di accordarsi esplicitamente con il rivenditore (*prima* dell'acquisto) per provare il programma con la propria stampante. L'interfaccia venduta con Amiprint funziona perfettamente con il C/64 Emulator, e in questo caso riesce a utilizzare anche le stampanti che non funzionano con Amiprint.

Misteri dei computers e dei programmi e cavi fatti artigianalmente !

```
100 printchr$(147)"caricatore routine di conversione ascii"
110 print"by spataro valentino":print
120 print"sintassi:":print"sys loc,so$,devso,devde,de$,mode"
130 print:print"dove:":print"so$, de$ nome file sorgente e destinaz."
140 print"devso, devde il numero di device"
150 print"mode=0 da c64 ad amiga":print"mode<>0 da amiga a c64"
160 print"es.:":print"sys 49152,filec64,8,10,fileamiga,0":print
170 sa=49152:input"start address (defaut=49152)";sa:sx=sa
180 b=0:ck=0
190 read a:if a<0 then b=b+1:read sm:if ck<>sm thenprint"errore in gruppo"b:end
200 if a=-1then ck=0:goto 190
210 if a=-2 then goto 230
220 pokesx,a:sx=sx+1:ck=ck+a:goto 190
230 pokesa+193,(sa+237)/256:pokesa+192,sa+237-peek(sa+193)*256:rem riloca unico
240 rem fine caricamento routine lm in memoria a partire da sa
250 x$="":fora=1to10:x$=x$+chr$(17):next:printchr$(147)
260 printchr$(19)x$:so=8:input "device sorgente (8)";so
270 so$="d":input"nome file sorgente (d)";so$:if so$<>"d" goto 350
280 p$="":input "parziale (* verra' aggiunto) ";p$:p$=p$+"*"
290 open8,so,0,"$:"+p$:get#8,a$,a$:printchr$(147)"premi f1 per terminare":b$=""
300 get#8,a$,a$
310 if a$=""or b$=chr$(133)thenclose8:print"premi":wait198,1:poke198,0:goto 250
320 get#8,b$,l$:ln$=str$(asc(b$+chr$(0))+asc(l$+chr$(0))*256):printtab(5)ln$" ";
330 get#8,a$:printa$;:getb$:ifa$<>"" and b$<>chr$(133) goto 330
340 print:goto300
350 st$="p":input "tipo file sorgente: p,s,r,u (p)";st$
360 de$="mom":input "file destinazione (mom)";de$
370 dt$="s":input "tipo file destinazione: p,s,r,u (s)";dt$
380 de=3:input "device destinaz. (3)";de
```

```
390 co=1:input "1 converte; 0 copia (1)";co
400 ifco=0then pokesa+191,44:print:goto 430:rem codice istruz. bit (v. art)
410 pokesa+191,32:rem mette il codice di istruzione jsr a rout di conversione
420 mode=0:input"1=da c64 a amiga;0=contrario (0)";mode
430 print"dati esatti ?  "chr$(157);:rem due spazi dopo ?
440 geta$:printa$;:ifa$="n"goto260
450 ifa$<>"s"goto440
460 print:print"premi ctrl per terminare lavoro durante il caricamento"
470 so$=so$+","+st$+",r":de$=de$+","+dt$+",w"
480 sys sa,so$,so,de,de$,mode:rem di tutto il lavoro si occupa la routine
490 if de=3 then print"premi un tasto":wait198,1:poke198,0:printchr$(147)
500 goto 260
19999 rem gruppo 1
20000 data 032,253,174,032,139,176,160,000,177,071,208,001,096,133,189
20010 data 200,177,071,133,253,200,177,071,133,254,032,241,183,134,182
20020 data 032,241,183,142,053,003,032,253,174,032,139,176,160,000,177
20030 data 071,208,001,096,141,052,003,200,177,071,133,251,200,177,071
20040 data 133,252,032,241,183,142,054,003,165,189,166,253,164,254,032
20050 data 189,255,169,003,166,182,160,003,032,186,255,032,192,255,169,-1,12442
20059 rem gruppo 2
20060 data 016,133,253,169,039,133,254,160,000,173,141,002,201,004,240
20070 data 117,032,183,255,201,000,208,023,162,003,032,198,255,032,207
20080 data 255,153,000,004,145,253,200,192,000,208,224,230,254,024,144
20090 data 219,230,254,165,254,141,059,003,140,058,003,162,003,032,195
20100 data 255,169,016,133,253,169,039,133,254,173,052,003,166,251,164
20110 data 252,032,189,255,169,004,174,053,003,160,007,032,186,255,032,-1,12063
20115 rem gruppo 3
20120 data 192,255,162,004,032,201,255,160,000,177,253,032,237,192,032
20130 data 210,255,200,192,000,208,002,230,254,165,254,205,059,003,048
20140 data 234,165,253,204,058,003,048,227,169,004,032,195,255,169,003
20150 data 032,195,255,169,004,032,195,255,032,204,255,096,174,054,003
20160 data 224,000,240,048,201,191,048,006,201,224,016,002,233,096,201
20170 data 094,048,007,201,126,016,003,233,030,096,201,031,048,005,201
20180 data 065,016,001,096,201,064,048,007,201,091,016,003,105,032,096,-1,12756
20185 rem gruppo 4
20190 data 201,126,048,002,169,064,096,201,013,208,001,096,201,010,208
20200 data 003,169,013,096,201,032,208,001,096,201,031,016,003,169,000
20210 data 096,201,097,048,007,201,123,016,003,233,031,096,201,065,048
20220 data 007,201,091,016,003,105,128,096,201,126,016,001,096,169,164
20230 data 096,-2,5865
20240 end

ready.
```

PIXMATE, PIU' CHE UN "PACCHETTO", UN VERO "SACCO" DI SORPRESE

Un pacchetto grafico che si differenzia dagli altri, pur straordinari, package per Amiga; consente un "trattamento" dell'immagine davvero sofisticato

di Luigi Callegari

Pixmate V1.1 è un prodotto della *Progressive Peripherals* veramente interessante, se si considera che è stato sviluppato per il trattamento *scientifico* delle immagini.

E' stato scritto da *Justin McCormick*, nientemeno che un collaboratore della *NASA*, sfruttando alcuni studi fatti per migliorare, via software, le immagini provenienti dalle sonde spaziali.

Le funzioni di PIXmate sono molto numerose e perfino il manuale originale, di circa centocinquanta pagine, rimanda, per approfondimenti tecnici, ad una bibliografia scientifica.

Qui vedremo, ordinatamente, il significato di tutte le funzioni di base e dei menu, dando qualche cenno per un uso immediato e pratico del pacchetto.

Menu Project

Il primo menu, come consuetudine di molti programmi Amiga, riguarda le funzioni di caricamento e registrazione dei files grafici da elaborare.

Le funzioni *Load Iff* e *Save Iff* hanno un significato evidente, consentendo di caricare e scrivere files dalla memoria di massa (Ram Disk, floppy, disco rigido). Le procedure di caricamento e salvataggio avvengono sempre tramite un requester speciale, piuttosto comodo. In esso compaiono a scorrimento i nomi di tutti i files della directory corrente: per scegliere il file da caricare / salvare basta clickare con il mouse sulla sua striscia, oppure digitarne il nome accanto al requester *File*, dopo avere correttamente specificato la *Path* nell'apposito requester che gli compare accanto. Ad esempio la stringa...

df0: Rag / Monica

...specifica il file chiamato *Monica* nella directory *Rag*, assunta presente nel dischetto inserito nel drive interno *(df0:)* di Amiga. Clickando sul gadget *Cancel!* si annulla l'operazione, mentre con *Last Dir* si ritorna alla directory precedente all'attuale.

Sono anche previsti gadget per cambiare il device di caricamento (*Df0:, Ram:, Dh0:...*) ed altri che consentono di avere visualizzati sullo schermo di caricamento i nomi dei file ordinati alfabeticamente (*Alpha*), per dimensione (*Size*) o per data di aggiornamento (*Date*).

La funzione *Compress* del menu *Project* consente di attivare *(On)* o disattivare *(Off)* la compressione automatica del

DOMANDE SPECIFICHE

In Redazione giungono sempre più spesso lettere con richieste di informazioni specifiche sull'uso dei programmi e dei pacchetti software per Amiga. I motivi di queste difficoltà da parte degli utenti è piuttosto ovvio: i manuali sono spesso carenti, o scritti in una lingua straniera sconosciuta.

Talvolta risultano troppo voluminosi e complicati da studiare, dal momento che spesso gli americani amano l'approccio di tipo "*tutorial*" (adatto per gli incompetenti, ma ostico per chi può muovere i primi passi da solo) per sapere come svolgere soltanto una certa funzione. Altre volte, invece, i manuali forniscono informazioni complete su argomenti complessi, ma tralasciano di spiegare come ottenere funzioni semplici.

La Redazione di C.C.C. si offre di aiutarvi. Scriveteci indicando ***chiaramente*** quali problemi pratici avete, per qualunque tipo di pacchetto Amiga (purchè questo sia "ragionevolmente" diffuso tra gli utenti): Deluxe Paint, Photon Paint, Pixmate, Sculpt 4D, GFA Basic, Lattice C, eccetera. Ovviamente risponderemo sulle pagine della rivista, nei limiti del possibile, privilegiando quelle domande che riteniamo possano essere più comuni tra gli utenti di Amiga. Suggeriamo di scrivere testi che possano essere pubblicati senza troppi "tagli", direttamente sulle pagine della nostra rivista..

file prima del salvataggio su disco secondo lo standard *Iff*. In ambedue i casi il file sarà ricaricabile con l'opzione *Load Iff* (ed anche da qualunque altro pacchetto grafico per Amiga), ma, consentendone la compressione, la sua dimensione sarà tipicamente inferiore del 20% - 30%. La funzione Read consente di scegliere il formato del file di caricamento: *Palette* (solo tavolozza cromatica), *Raw Format* (formato a bitplanes puri), *Digiview 1.0* (formato del vecchio digitalizzatore *NewTek*) e *Neochrome* (formato usato da programmi grafici per Atari ST). Per caricare un file di formato diverso da *Iff* è dunque necessario prima indicare in questo sottomenu il suo formato, poi scegliere *Load Iff*.

La funzione *Write* consente di scrivere il file grafico attuale in uno tra due formati differenti *(Palette* o *Raw Format)*. La funzione *Delete* consente di cancellare un file, tramite il requester usato anche per il caricamento ed il salvataggio dei files.

La funzione *Quit* termina l'esecuzione del programma PIXmate.

Menu Edit

Le funzioni del secondo menu di PIXmate consentono la manipolazione delle immagini secondo modalità semplici.

L'opzione *Undo Changes*, ottenibile anche premendo semplicemente il tasto **u**, annulla gli effetti dell'ultima operazione eseguita, purchè si sia previsto il cosiddetto *buffer di Undo*, ovvero memoria permettendo.

Le opzioni *Flip Other*, *Copy Other*, *Clip Other* chiamano in gioco il cosiddetto *Other Screen* (lo "schermo ombra", definizione non inventata da Occhetto).

Si tratta di un'area alternativa, supplementare a quella di *Undo*, gestita tramite le tre funzoni che consentono, rispettivamente, di scambiare l'area principale con l'alternativa, di copiare l'area attuale nell'alternativa e di effettuare il ritaglio di una porzione indicata con il mouse nell'area visualizzata trasferendola in quella alternativa.

Le funzioni *Toggle Title* e *Toggle Sprite* servono, rispettivamente, per disabilitare la barra di intestazione dello schermo e permettere di contemplare per intero l'immagine, e lo sprite del mouse.

Con *Recenter* si ricentra lo schermo, eventualmente fatto scorrere tramite i tasti di spostamento cursore, mentre con *ClearArea* e *ClearScreen* si puliscono, rispettivamente, un'area indicata sullo schermo e lo schermo intero.

Le funzioni *Kill Other*, *Kill Undo* e *Disable Undo* servono, rispettivamente, per annullare lo schermo ombra, l'area di Undo e per disabilitare la funzione di Undo dopo ogni operazione.

Tutte consentono risparmi di memoria.

Menu Colors

Le funzioni del terzo menu di PIXmate consentono, come prevedibile, di manipolare i colori, ovvero la tavolozza cromatica *(palette)* del disegno visualizzato.

Con *Cycle* si produce un eventuale scorrimento ciclico dei colori, come predisposto tramite apposite scalature dei colori nella tavolozza (funzione *Spread*, presente anche in **Deluxe Paint**), mentre con Palette si visualizza un requester vagamente simile a quello di Deluxe Paint, che consente di manipolare le componenti cromatiche su scala *RGB* (Rosso, Verde, Blu) oppure *HSV* (Tinta, Saturazione e Valore) tramite appositi slider, sempre secondo modalità ben note a chi ha già lavorato con programmi tipo Deluxe Paint.

La funzione *Pack Colors* esegue un'analisi della pagina grafica assegnando i colori più usati ai primi registri di Denise e lasciando agli ultimi i colori meno usati (ciò non funziona per il modo HAM, generato algoritmicamente).

In questo modo è possibile sia risparmiare un poco di spazio nella memorizzazione della pagina grafica su disco, sia eventualmente di tagliare con più efficacia il numero massimo di colori (bitplanes) assegnati al disegno tramite la funzione *Less Colors*.

La funzione *Sort Colors* riordina i colori assegnati ai registri hardware in modo ascendente o discendente (da sub - opzione) secondo il loro utilizzo. La funzione Less Colors consente di spe-

cificare esattamente quanti colori si vogliono sul disegno, da 2 a 64 (modo *Extra Halfbrite*).

Si noti che se si ha sul video un disegno a 32 colori, e se ne chiedono 31, il colore assegnato al registro numero 30 (la numerazione hardware parte da zero) viene ricopiato anche in posizione 31, ma durante la compressione per il salvataggio esso non sarà considerato ed il file potrà presumibilmente essere più corto.

Si ricordi anche che il numero di colori è determinato dalla formula:

colori = 2 elevato a bitplanes

quindi con 5 bitplanes si hanno comunque 32 colori, mentre con 4 bitplanes 16, e con un solo bitplane due colori.

La funzione *Extract* consente la cosiddetta scomposizione cromatica del disegno. Serve essenzialmente a chi deve produrre files per unità di fotocomposizione e stampa, così come la funzione *Automerge*, che è in grado di ricostruire un file colorato normalmente sul video partendo dai files singoli che contengono le singole componenti cromatiche.

Le tre funzioni *Match Palette*, *Copy Palette* e *Swap Palette* consentono di assegnare al grafico attuale la tavolozza cromatica assegnata anche ad un file su disco (leggendola dopo avere indicato il file tramite il solito requester di I/O), copiare una tavolozza cromatica e scambiarla.

Le funzioni *Complement* e *Pseudo Colors* modificano la tavolozza del disegno attualmente visualizzato assegnando, rispettivamente, i colori complementari ed una serie di tinte casuali (diverse ogni volta).

Menu Effects

Nel quarto menu sono presenti le funzioni di elaborazione delle immagini più sofisticate. La funzione *Display* fa comparire un requester che consente di eseguire parecchie operazioni sulla pagina grafica visualizzata: ridurla per il largo *(Thinner)* o per il lungo *(Shorter)*, invertirla orizzontalmente o verticalmente *(Flip X, Flip Y)*, cambiarne lo stato ed il modo *(ViewMode)*, alterarne le dimensioni specificando tramite requester l'esatta risoluzione orizzontale e verticale, ingrandirla orizzontalmente *(Wider)* o verticalmente *(Taller)*.

E' anche possibile, tramite mouse e gadget, ruotare o scambiare i singoli bitplanes.

Il requester che compare scegliendo *Image Process* comprende tutte le più sofisticate e scientifiche funzioni di manipolazione dell'immagine, per il quale diventa indispensabile consultare il manuale di istruzioni, che destina molte pagine alla loro spiegazione.

Diremo soltanto che le operazioni eseguibili sul disegno si dividono in due categorie: orientate al singolo pixel ed orientate al blocco di pixel. Oltre a queste si possono eseguire le normali operazioni logiche sui pixel: *And, Eor, Or, Not* e *Sub*.

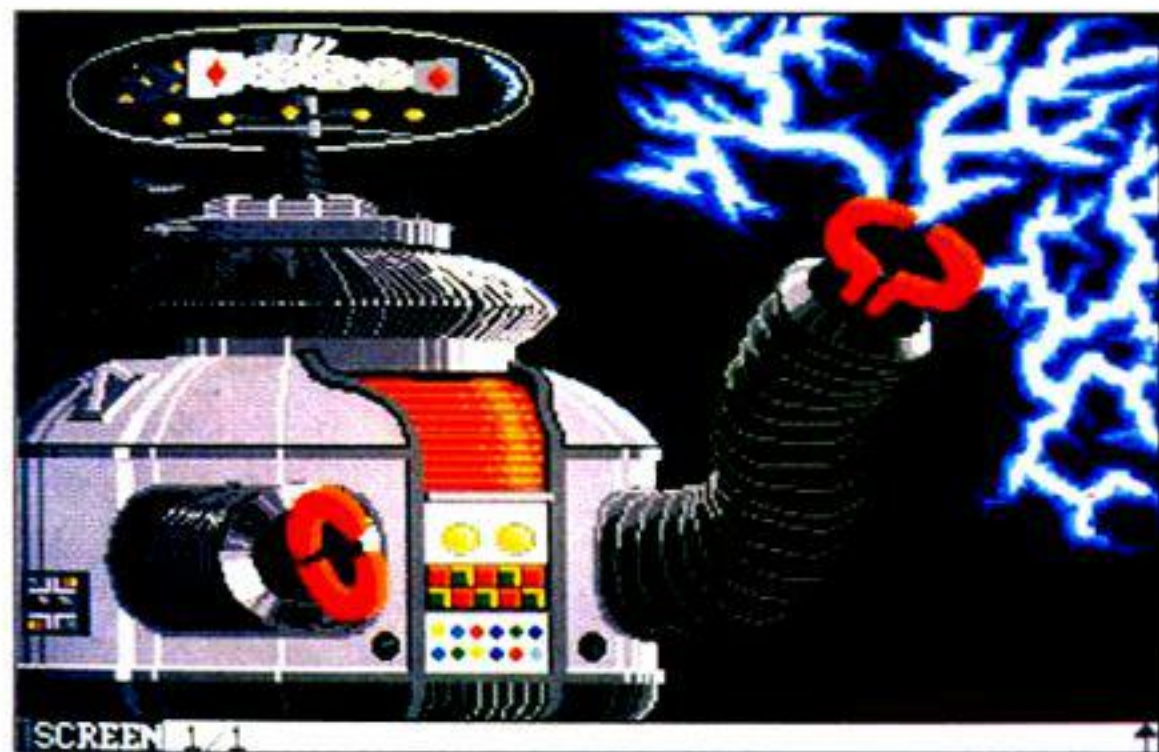
Le operazioni orientate agli insiemi di pixel si attivano tramite gli appositi gadget: *Avg* esegue la media della serie di tre pixel orizzontali e li sostituisce col valore calcolato, *Rnd* sostituisce il colore del pixel centrale in un quadrato 3 x 3 con un valore scelto casualmente tra gli otto pixel circostanti. *MF1* e *MF2* assegnano al pixel di centro di un blocco 3 x 3 oppure 5 x 5, rispettivamente, il valore medio calcolato per i pixel della matrice circostante. *Usm* esegue un sofisticato algoritmo chiamato "correlazione Laplaciana", che tenta di ottenere una immagine più nitida sottraendo, all'immagine originale, una immagine "diffusa" automaticamente. *Lce* rinforza i piccoli dettagli del grafico (la NASA le usò per cercare tracce di canali, vulcani e marziani nelle immagini delle superfici planetarie inviate dalle sonde spaziali): si calcola il valore medio di una matrice di venticinque pixel, poi la si esamina col pixel centrale e se è più chiaro (rispetto ad un limite indicato tramite lo slider *Threshold*) viene rischiarato, mentre se è più scuro viene scurito.

La funzione *Histograph* del menu *Effects* consente di ottenere un grafico ad istogramma verticale delle ricorrenze dei colori nel disegno, e di "equalizzarli" tramite appositi slider.

Le funzioni *Ham To 32* e *Ham To 64*, come prevedibile, convertono una pagina dal formato *Hold and Modify* (4096 colori) ad una normale pagina con 32 colori normali o 64 in *EHB*.

La funzione *Reformat As* consente di cambiare rapidamente la risoluzione della pagina grafica attuale secondo gli standard PAL.

Menu Info

Le quattro funzioni dell'ultimo menu di PIXmate consentono di avere varie statistiche sulla pagina grafica attuale. La funzione *Memory* indica il consumo di memoria, *Format* il formato ed il numero di colori usati, *Count* il numero di pixel dei vari colori sullo schermo e *Coords* le coordinate del mouse in tempo reale.

QUANTO COSTA IL TUO COMMODORE

Amiga 2000 - L. 2.715.000

Microprocessore Motorola MC68000 - Clock 7.16MHz - Kickstart ROM - Memoria RAM: 1 MByte - 3 chip custom per DMA, Video, Audio, I/O - 5 Slot di Espansione Amiga Bus 100 pin Autoconfig™ - 1 Slot di Espansione 86 pin per Schede Coprocessore - 2 Slot di Espansione compatibili AT/XT - 2 Slot di Espansione compatibili XT - 2 Slot di Espansione Video - 1 Floppy Disk Drive da 3 1/2", 880 KBytes - Porta seriale RS232C - Sistema Operativo single-user, multitasking AmigaDOS - Compatibilità MS-DOS XT/AT disponibile con schede interne Janus (A2088 - A2286) - Monitor escluso

Amiga 500 - L. 995.000

Microprocessore Motorola MC68000 - Clock 7.16 MHz - Kickstart ROM - Memoria RAM: 512 KBytes - 3 Chip custom per DMA, Video, Audio, I/O - 1 Floppy Disk Driver da 3 1/2", 880 KBytes - Porta seriale RS232C - Porta parallela Centronics

Videomaster 2995 - L. 1.200.000

Desk Top Video - Sistema per elaborazini video semiprofessionale composto da genlock, digitalizzatore e alloggiamento per 3 drive A2010 - Ingressi videocomposito (2), RGB - Uscite Videocomposito, RF, RGB + sync -

Floppy Disk Driver A 1010 - L. 335.000

Floppy Disk Driver - Drive esterno da 3 1/2" - Capacità 880 KBytes - Collegabile a tutti i modelli della linea Amiga, alla scheda A2088 e al PC1

Floppy Disk Drive A 2010 - L. 280.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" - Capacità 880 KBytes - Collegabile ad Amiga 2000

Hard Disk A 590 - L. 1.750.000

Hard Disk+Controller+RAM - Scheda Controller - Hard Disk da 3 1/2" 20 MBytes - 2 MBytes "fast" RAM - Collegabile all'Amiga 500

Scheda Janus A 2088 + A 2020 - L. 1.050.000

Scheda Janus XT+Floppy Disk Drive da 5 1/4", 360 KBytes - Scheda Bridgeboard per compatibilità MS-DOS (XT) in Amiga 2000 - Microprocessore Intel 8088 - Coprocessore matematico opzionale Intel 8087

A2286+A2020 - L. 1.985.000

Scheda Janus AT+Floppy Disk Drive da 5 1/4", 1.2 MBytes - Scheda Bridgeboard per compatibilità MS-DOS (AT) in Amiga 2000 - Microprocessore Intel 80287 - Clock 8 MHz - RAM: 1 MBytes on-board - Floppy Disk Controller on-board - Floppy Disk Driver disegnato per l'installazione all'interno dell'Amiga 2000 -

Scheda A2620 - L. 2.700.000

Scheda Processore Alternativo 32 bit - Scheda per 68020 e Unix - Microprocessore Motorola MC68020 - Coprocessore matematico Motorola MC68881 (opzionale MC68882)

Scheda A Unix - L. 3.250.000

Sistema Operativo AT&T Unix System V Release 3 - Per Amiga 2000 con scheda A2620 e Hard Disk 100 MBytes

Hard Disk A2092+PC5060 - L. 1.020.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+2092 - L. 1.240.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+A2094 - L. 1.900.000

Stesse caratteristiche del kit A2092 ma con disco da 40 MBytes

Espansione di memoria A2058 - L. 1.149.000

Espansione di memoria - Scheda di espansione per Amiga 2000 - Fornita con 2 MBytes "fast" RAM, espandibile a 4 o 8 MBytes

Scheda Video A2060 - L. 165.000

Modulatore video - Scheda modulatore video interna per Amiga 2000 - Uscite colore e monocromatica - Si inserisce nello slot video dell'Amiga 2000

Genlock Card A2301 - L. 420.000

Genlock - Scheda Genlock semiprofessionale per Amiga 2000 - Permette di miscelare immagini provenienti da una sorgente esterna con immagini provenienti dal computer

Professional Video Adapter Card A2351 - L. 1.500.000

Professional Video Adapter - Scheda Video Professionale per Amiga 2000 (B) - Genlock qualità Broadcast - Frame Grabber - Digitalizzatore - Include software di controllo per la gestione interattiva
(Disponibile da maggio '89)

A501 - L. 300.000

Espansione di memoria - Cartuccia di espansione di memoria da 512 KBytes per A500

A520 - L. 45.000

Modulatore RF - Modulatore esterno A500 - Permette di connettere qualsiasi televisore B/N o colori ad Amiga 500

A Scart - L. 27.000

Cavo di collegamento A500/A2000 con connettore per televisione SCART

Monitor a colori 1084 - L. 595.000

Monitor a colori ad alta risoluzione - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor a colori 2080 - L. 770.000

Monitor a colori ad alta risoluzione e lunga persistenza - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Frequenza di raster 50 Hz - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor Monocromatico A2024 - L. 1.235.000

Monitor monocromatico a fosfori "bianco-carta" - Turbo 14" antiriflesso - (Disponibile da marzo '89)

PC60/40 - L. 7.812.000

Microprocessore Intel 80386 - Coprocessore matematico opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 funzioni - Sistema Operativo MS-DOS 3 2.1 - Interprete GW-Basic

PC60/40C - L. 8.127.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 60/80 - L. 10.450.000

Microprocessore Intel 80386 - Coprocessore opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Memoria RAM: 2.5 MBytes - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Floppy Disk Drive opzionale da 3 1/2", 1.44 MBytes - 1 Hard Disk da 80 MBytes - 2 Porte parallele Centronics - Mouse video EGA (compatibile MDA - Hercules - CGA). Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Ambiente Operativo Microsoft Windows/386 - Interprete GW-Basic

PC60/80C - L. 10.700.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC40/20 - L. 4.100.000

Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/20C - L. 4.350.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 40/40 - L. 5.285.000

Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/40C - L. 5.535.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

1352 - L. 78.000

Mouse - Collegabile con Microsoft Bus Mouse - Collegabile direttamente a PC1, PC10/20 - III, PC40 - III

PC910 - L. 355.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" per PC10/20-I-II-III - Capacità 360 o 720 KBytes selezionabile tramite "config. sys" - Corredo di telaio di supporto per l'installazione in un alloggiamento per un drive da 5 1/4" - Interfaccia identica ai modelli da 5 1/4"

PC1 - L. 995.000

Microprocessore Intel 8088 - 1 Floppy Disk Drive da 5 1/4" - Porta seriale RS232C - Porta parallela Centronics - - Monitor monocromatico 12" - Tastiera 84 tasti - Sistema Operativo MS-DOS 3.2 - Interprete GW-Basic

PCEXP1 - L. 640.000

PC Expansion Box - Box esterno di espansione per PC 1 - Alimentatore aggiuntivo incluso - Contiene 3 Slot di Espansione compatibili Ibm XT - Alloggiamento per Hard Disk da 5 1/4" - Si posiziona sotto il corpo del PC1 e viene collegato tramite degli appositi connettori

PC10-III - L. 1.360.000

Microprocessore Intel 8088 Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - Memoria RAM: 640 KBytes - 2 Floppy Disk Drive da 5 1/4", 360 KBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC10-IIIC - L. 1.675.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

PC20-III - L. 2.095.000

Microprocessore Intel 8088 - Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - 1/4", 360 KBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse) - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC20-IIIC - L. 2.410.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

Nuovo C64 - L. 325.000
Nuovo Personal Computer CPU 64 KBytes RAM - Vastissima biblioteca software disponibile - Porta seriale Commodore - Porta registratore per cassette - Porta parallela programmabile -

C128D - L. 895.000
Personal Computer CPU 128 KBytes RAM espandibile a 512 KBytes - ROM 48 KBytes - Basic 7.0 - Tastiera separata - Funzionante in modo 128,64 o CP/M 3.0 - Include floppy disk drive da 340 KBytes

Floppy Disk Drive 1541 II - L. 365.000
Floppy Disk Drive - Floppy Disk Drive da 5 1/4" singola faccia - Capacità 170 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

Floppy Disk Dirve 1581 - L. 420.000
Floppy Disk Drive da 3 1/2" doppia faccia - Capacità 800 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

1530 - L. 55.000
Registratore a cassette per C64, C128, C128D

Accessori per C64 - 128D

1700 - Espansione di memoria - Cartuccia di espansione di memoria a 128 KBytes per C128 **- L. 170.000**

1750 - Espansione di memoria - Cartuccia di espansione di memoria 512 KBytes per C128 - **L. 245.000**

1764 - Espansione di memoria - Cartuccia di espansione di memoria a 256 KBytes per C64 - Fornita di alimentatore surdimensionato - **L. 198.000**

16499 - Adattatore Telematico Omologato - Collegabile al C64 - Permette il collegamento a Videotel, P.G.E. e banche dati **L. 149.000**

1399 - Joystick - Joystick a microswitch con autofire - **L. 29.000**

1351 - Mouse - Mouse per C64, C128, C128D - **L. 72.000**

Monitor Monocromatico 1402 - L. 280.000
Monitor monocromatico a fosfori "bianco-carta" - Turbo 12" antiriflesso - Ingresso TTL - Compatibile con tutta la gamma PC

Monitor Monocromatico 1404 - L. 365.000
Monitor monocromatico a fosfori ambra - Turbo 14" antiriflesso a schermo piatto - Ingresso TTL - Compatibile con tutta la gamma PC - Base orientabile

Monitor Monocromatico 1450 - L. 470.000
Monitor monocromatico BI-SYNC a fosfori "bianco-carta" - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Monitor a colori 1802 - L. 445.000
Monitor a colori - Turbo 14" - Collegabile a C64, C128, C128D

Monitor monocromatico 1900 - L. 199.000
Monitor monocromatico a fosfori verdi - Turbo 12" antiriflesso - Ingresso videocomposito - Compatibile con tutta la gamma Commodore

Monitor a colori 1950 - L. 1.280.000
Monitor a colori BI-SYNC alta risoluzione - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Stampante MPS 1230 - L. 465.000
Stampante a matrice di punti - Testina a 9 aghi - 120 cps - Bidirezionale - 80 colonne - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

MPS 1230R - L. 19.000
Nastro per stampante

Stampante MPS 1500C - L. 495.000
Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia parallela Centronics - Compatibile con la gamma Amiga e PC

MPS1500R - L. 37.000
Nastro a colori per stampante

MPS1500R - L.37.000
Nastro a colori per stampante

Stampante MPS 1550C - L. 575.000
Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

LOMBARDIA

Milano
• AL RISPARMIO - V.LE MONZA 204
• BCS - VIA MONTEGANI 11
• BRAHA A. - VIA PIER CAPPONI 5
• E.D.S. - C.SO PORTA TICINESE 4
• FAREF - VIA A. VOLTA 21
• FLOPPERIA - V.LE MONTENERO 31
• GBC - VIA CANTONI 7 - VIA PETRELLA 6
• GIGLIONI - V.LE LUIGI STURZO 45
• L'UFFICIO 2000 - VIA RIPAMONTI 213
• LOGITEK - VIA GOLGI 60
• LU - MEN - VIA SANTA MONICA 3
• MARCUCCI - VIA F.LLI BRONZETTI 37
• MELCHIONI - VIA P. COLLETTA 37
• MESSAGGERIE MUSICALI - GALLERIA DEL CORSO 2
• NEWEL - VIA MAC MAHON 75
• PANCOMMERZ ITALIA - VIA PADOVA 1
• SUPERGAMES - VIA VITRUVIO 38
• 68000 E DINTORNI - VIA WASHINGTON 91
Provincia di Milano
• GINO FERRARI CENTRO HI-FI - VIA MADRE CABRINI 44 - S. ANG. LODIGIANO
• F.LLI GALIMBERTI - VIA NAZIONALE DEI GIOVI 28/36 - BARLASSINA
• TECNOLUX - VIA PIETRO NENNI 5 - BERNATE TICINO
• OGGIONI & C. - VIA DANTE CESANA 27 - CARATE BRIANZA
• AL RISPARMIO - VIA U. GIORDANO 57 - CINISELLO BALSAMO
• GBC - V.LE MATTEOTTI 66 - CINISELLO BALSAMO
• CASA DELLA MUSICA - VIA INDIPENDENZA 21 - COLOGNO MONZESE
• PENATI - VIA VERDI 28/30 - CORBETTA
• EPM SYSTEM - V.LE ITALIA 12 - CORSICO
• P.G. OSTELLARI - VIA MILANO 300 - DESIO
• CENTRO COMPUTER PANDOLFI - VIA CORRIDONI 18 - LEGNANO
• COMPUTEAM - VIA VECELLIO 41 - LISSONE
• M.B.M. - C.SO ROMA 112 - LODI
• L'AMICO DEL COMPUTER - VIA CASTELLINI 27 - MELEGNANO
• BIT 84 - VIA ITALIA 4 - MONZA
• IL CURSORE - VIA CAMPO DEI FIORI 35 - NOVATE MIL.
• I.C.O. - VIA DEI TIGLI 14 - OPERA
• R & C ELGRA - VIA SAN MARTINO 13 - PALAZZOLO MIL.
• ESSEGIEMME SISTEMI SAS - VIA DE AMICIS 24 - RHO
• TECNO - CENTRO - VIA BARACCA 2 - SEREGNO
• NIWA HARD&SOFT - VIA B. BUOZZI 94 - SESTO SAN GIOV.
• COMPUTER SHOP - VIA CONFALONIERI 35 - VILLASANTA
• ACTE - VIA B. CREMIGNANI 13 - VIMERCATE
• IL COMPUTER SERVICE SHOP - VIA PADANA SUPERIORE 197 - VIMODRONE
Bergamo
•D.R.B. - VIA BORGO PALAZZO 65
• TINTORI ENRICO &C. - VIA BROSETA 1
• VIDEO IMMAGINE - VIA CARDUCCI c/o CITTA' DI MERCATO
Provincia di Bergamo
BERTULEZZI GIOVANNI - VIA FANTONI 48 - ALZANO LOMBARDO
• COMPUTER SHOP - VIA VITTORIO VENETO 9 - CAPRIATE SAN GERVASIO
• B.M.R. - VIA BUTTARO 4/T - DALMINE
• MEGABYTE 2 - VIA ROMA 61/A - GRUMELLO
• OTTICO OPTOMETRISTA ROVETTA - P.ZZA GARIBALDI 6 - LOVERE
• COMPUTER POINT - VIA LANTIERI 52 - SARNICO
• A.B. INFORMATICA - STRADA STATALE CREMASCA 66 - URGNANO
Brescia
• MASTER INFORMATICA - VIA F.LLI UGONI 10/B
PROVINCIA DI BRESCIA
• MISTER BIT - VIA MAZZINI 70 - BRENO
• CAVALLI PIETRO - VIA 10 GIORNATE 14 BIS - CASTREZZATO
• VIETTI GIUSEPPE - VIA MILANO 1/B - CHIARI
• MEGABYTE - P.ZZA MALUEZZI 14 - DESENZANO DEL GARDA
• BARESI RINO &C. - VIA XX SETTEMBRE 7 - GHEDI
• INFO CAM - VIA PROVINCIALE 3 - GRATACASOLO
• "PAC-LAND" di GARDONI - CENTRO COM.le - LA CASA DI MARGHERITA D'ESTE - VIA GIORGIONI 21
Como
• IL COMPUTER - VIA INDIPENDENZA 90
• 2M ELETTRONICA - VIA SACCO 3
Provincia di Como
• ELTRON - VIA IV NOVEMBRE 1 - BARZANO'
• DATA FOUND - VIA A. VOLTA 4 -ERBA
• CIMA ELETTRONICA - VIA L. DA VINCI 7 - LECCO
• FUMAGALLI - VIA CAIROLI 48 -LECCO
• RIGHI ELETTRONICA - VIA G. LEOPARDI 26 - OLGIATE COMASCO
Cremona
• MONDO COMPUTER - VIA GIUSEPPINA 11/B
• PRISMA - VIA BUOSO DA DOVARA 8
• TELCO - P.ZZA MARCONI 2/A
Provincia di Cremona
• ELCOM - VIA IV NOVEMBRE 56/58 - CREMA
• EUROELETTRONICA - VIA XX SETTEMBRE 92/A - CREMA
Mantova
• COMPUTER CANOSSA - GAL. FERRI 7
• 32 BIT - VIA C. BATTISTI 14
• ELET. di BASSO - V.LE RISORGIMENTO 69
Provincia di Mantova
• CLICK - ON COMPUTER - S.S. GOITESE 168 - GOITO
Pavia
• POLIWARE - C.SO C. ALBERTO 76
• SENNA GIANFRANCO - VIA CALCHI 5
Provincia di Pavia
• A. FERRARI - C.SO CAVOUR 57 - MORTARA
• LOGICA MAINT - V.LE M.TE GRAPPA 32 - VIGEVANO
• M. VISENTIN - C.SO V. EMANUELE 76 - VIGEVANO
Sondrio
• CIPOLLA MAURO - VIA TREMOGGE 25
Provincia di Sondrio
• FOTONOVA - VIA VALERIANA 1 - S.PIETRO DI BERBENNO
Varese
• ELLE - EFFE - VIA GOLDONI 35
• IL C.TRO ELET. - VIA MORAZZONE 2
• SUPERGAMES - VIA CARROBBIO 13
Provincia di Varese
• BUSTO BIT - VIA GAVINANA 17-BUSTO A.
• MASTER PIX - VIA S.MICHELE 3 - BUSTO A.
• PUNTO UFFICIO - VIA R.SANZIO 8 - GALLARATE
• GRANDI MAGAZZINI BOSSI - VIA CLERICI 196 - GERENZANO
• J.A.C. - C.so MATTEOTTI 38 - SESTO C.

PIEMONTE
Alessandria
• BIT MICRO - VIA MAZZINI 102
• SERV. INFOR. - VIA ALESSANDRO III 47
Provincia di Alessandria
• SONY ITALIANA - VIA G. MANARA 7 - CASALE MONFERRATO
• SGE ELETTRONICA - VIA BANDELLO 19 - TORTONA
• COMPUTER TEMPLE - VIA F. CAVALLOTTI 13 - VALENZA
Asti
• ASTI GAMES - C.SO ALFIERI 26
• RECORD - C.SO ALFIERI 166/3 (Galleria Argenta)
Cuneo
• ROSSI COMPUTERS - C.SO NIZZA 42
Provincia di Cuneo
• PUNTO BIT - C.SO LANGHE 26/C - ALBA
• BOSETTI - VIA ROMA 149 - FOSSANO
• COMPUTERLAND - VIA MAZZINI 30/32 - SALUZZO
Novara
• PROGRAMMA 3 - V.LE BUONARROTI 8
• PUNTO VIDEO - C.so RISORGIMENTO 39/B
Provincia di Novara
• COMPUTER - VIA MONTE ZEDA 4 -ARONA
• ALL COMPUTER - C.SO GARIBALDI 106 - BORGOMANERO
• S.P.A. - C.SO DISSEGNA 21/BIS - DOMODOSSOLA
• ELLIOTT COMPUTER SHOP - VIA DON MINZONI 32 - INTRA
• TRISCONI VALERIA - VIA MAZZINI 90 - OMEGNA
Torino
• ABA ELETTRONICA - VIA C. FOSSATI 5/P
• ALEX COMPUTER E GIOCHI - C.SO FRANCIA 333/4
• COMPUTER HOME - VIA SAN DONATO 46/D
• COMPUTING NEW - VIA M. POLO 40/E
• C.D.M. ELETTR. - VIA MAROCHETTI 17
• DE BUG - C.SO V. EMANUELE II 22
• DESME UNIVERSAL - VIA S.SECONDO 95
• FDS ALTERIO - VIA BORGARO 86/D
• IL COMPUTER - VIA N. FABRIZI 126
• MICRONTEL - C.SO D. degli ABRUZZI 28
• PLAY GAMES SHOP - VIA C. ALBERTO 39/E
• RADIO TV MIRAFIORI - C.SO UNIONE SOVIETICA 381
• SMT ELETTRONICA - VIA BIBIANA 83/bis
Provincia di Torino
• PAUL E CHICO VIDEOSOUND - VIA V.EMANUELE 52 - CHIERI
• BIT INFORMATICA - VIA V. EMANUELE 154 - CIRIE'
• HI - FI CLUB - C.SO FRANCIA 92C - COLLEGNO
• MISTER PERSONAL - VIA CATTANEO 52 - FAVRIA
• I.C.S. - VIA TORINO 73 - IVREA
• DAG - VIA I MAGGIO 40 - LUSERNA S. GIOVANNI
• EUREX - C.SO INDIPENDENZA 5 - RIVAROLO CANAVESE
• DIAM INFORMATICA - C.SO FRANCIA 146 bis - RIVOLI
• FULLINFORMATICA - VIA V.VENETO 25 - RIVOLI
• GAMMA COMPUTER - VIA CAVOUR 3A-3B - SET.TORINESE
Vercelli
•ELETTROGAMMA - C.SO BORMIDA 27 ang. V.Montanara
• ELETTRONICA - STRADA TORINO 15
Provincia di Vercelli
• C.S.I. TEOREMA - VIA LOSANA 9 - BIELLA
• SIGEST - VIA BERTODANO 8 - BIELLA
• REMONDINO FRANCO - VIA ROMA 5 - BORGOSESIA
• FOTOSTUDIO TREVISAN - VIA XXV APRILE 24/B - COSSATO
• STUDIO FOTOGRAFICO IMARISIO - P.ZZA M. LIBERTA' 7 - TRINO

VENETO
Belluno
• UP TO DATE - VIA V. VENETO 43
Provincia di Belluno
• GUERRA COMPUTERS - V.LE MAZZINI 10/A - FELTRE
Padova
• BIT SHOP - VIA CAIROLI 11
• COMPUMANIA - VIA T. CAMPOSANPIERO 37
• D.P.R. DE PRATO R. - V.LO LOMBARDO 4
• G.F. MARCATO - VIA MADONNA DELLA SALUTE 51/53
• SARTO COMPUTER - VIA ARMISTIZIO 79
Provincia di Padova
• COMPUTER SERVICE - BORGO TREVISO 150 - CITTADELLA
Treviso
• BIT 2000 - VIA BRANDOLINI D'ADDA 14
• GUERRA EGIDIO & C. - V.LE CAIROLI 95
Provincia di Treviso
• DE MARIN COMPUTERS - VIA MATTEOTTI 142 - CONEGLIANO
• SIDESTREET - VIA SALVO D'ACQUISTO 8 - MONTEBELLUNA
• FALCON ELETTROAUDIOVIDEO - VIA TERRAGGIO 116 - PREGANZIOL
Venezia
• GUERRA EGIDIO & C. - VIA BISSUOLA 20/A - MESTRE
• TELERADIO FUGA - SAN MARCO 3457
Provincia di Venezia
• GUERRA EGIDIO & C. - VIA VIZZOTTO 29 - SAN DONA' DI PIAVE
• REBEL - VIA F. CRISPI 10 - SAN DONA' DI PIAVE
Verona
• CASA DELLA RADIO - VIA CAIROLI 10
• TELESAT - VIA VASCO DE GAMA 8
• Provincia di Verona
• UBER - CP 0363(RAG.SOC. DERTA) - VIA MASCAGNI 31 -CASTE' D'AZZANO
• FERRARIN - VIA DEI MASSARI 10 - LEGNAGO
• COMPUTERS CENTER - VIA CANTORE 26 - VILLAFRANCA
Vicenza
• ELET. BISELLO - V.LE TRIESTE 427/429
• SCALCHI MARKET - VIA C.' BALBI 139
Provincia di Vicenza
• SCHIAVOTTO - VIA ZANELLA 21 - CAVAZZALE
• GUERRA E. & C. - V.LE DELLE INDUSTRIE - MONTECCHIO MAGGIORE

FRIULI VENEZIA GIULIA
Gorizia
• E.CO. ELETTRONICA - VIA F.LLI COSSAR 23
Trieste
• AVANZO GIACOMO - P.ZZA CAVANA 7
• COMPUTER SHOP - VIA P. RETI 6
• COMPUTIGI - VIA XX SETTEMBRE 51
• CTI - VIA PASCOLI 4
Udine
• MOFERT 2 - VIA LEOPARDI 21
• R.T. SISTEM UDINE - VIA L. DA VINCI 99
Provincia di Udine
• IL PUNTO ELETTRONICO - VIA VENDRAMIN 184 - LATISANA
• IDRENO MATTIUSSI &C. - VIA LICINIANA 58 - MARTIGNACCO

TRENTINO ALTO ADIGE
Bolzano
• COMPUTER POINT - VIA ROMA 82/A
• MATTEUCCI PRESTIGE - VIA MUSEO 54
Provincia di Bolzano
• RADIO MAIR-ELECTRO - VIA CENTRALE 70 - BRUNICO
• ELECTRO RADIO HENDRICH - VIA DELLE CORSE 106 - MERANO
• ERICH KONTSCHIEDER - PORTICI 313 - MERANO
• ELECTRO TAPPEINER - P.ZZA PRINCIPALE 90 - SILANDRO
Trent0
• CRONST - VIA G. GALILEI 25
Provincia di Trento

• AL RISPARMIO - C.SO VERONA 138 - ROVERETO

LIGURIA
Genova
• ABM COMPUTER - P.ZZA DE FERRARI 24 rosso
• CAPRIOTTI G. - IA MAMIANI 4r - SAMPIERDARENA
• C.tro ELET. - VIA CHIARAVAGNA 10 R - VIA SESTRI 69R
• COM.le SOTTORIPA - VIA SOTTORIPA 115/117
• FOTOMONDIAL - VIA DEL CAMPO 3-5-9-11-13 r
• LA NASCENTE - VIA SAN LUCA 4/1
• PLAY TIME - VIA GRAMSCI 3/5/7 rosso
• RAPPR-EL - VIA BORGORATTI 23 R
Imperia
• CASTELLINO - VIA BELGRANO 44
Provincia di Imperia
• CENTRO HI-FI VIDEO - VIA DELLA REPUBBLICA 38 -SANREMO
• CASTELLINO - VIA GENOVA 48 - VENTIMIGLIA
La Spezia
• I.L. ELETTRONICA - VIA V. VENETO 123
Provincia di La Spezia
• I.L. ELETTRONICA - VIA AURELIA 299 - FORNOLA DI VEZZANO
Savona
• CASTELLINO - C.SO TARDY E BENECH 101
Provincia di Savona
• CELESIA ENZA - VIA GARIBALDI 144 - LOANO

EMILIA
Bologna
• EUROELETTRICA - VIA RANZANI 13/2
• MINNELLA ALTA FEDELTA' - VIA MAZZINI 146/2
• MORINI & FEDERICI - VIA MARCONI 28/C
• STERLINO - VIA MURRI 73/75
Provincia di Bologna
• S.C. COMPUTERS - VIA E. FERMI 4 - CASTEL SAN PIETRO
• S.P.E. INFORMATICA - VIA DI MEZZO PONENTE 385 - CREVALCORE
• ARCHIMEDE SISTEMI - VIA EMILIA 124 - S. LAZZARO DI SAVENA
Modena
• CO - EL - VIA CESARI 7
• ORSA MAGGIORE - P.ZZA MATTEOTTI 20
• VIDEO VAL WILLY COMPUTERS - VIA CANALETTO 223
Provincia di Modena
• NEW MEDIA SYSTEM - VIA ROMA 281 - SOLIERA
Parma
• BABARELLI G. - VIA B. PARENTE 14/A/B
Provincia di Parma
• PONGOLINI - VIA CAVOUR 32 - FIDENZA
Piacenza
• COMPUTER LINE - VIA G. CARDUCCI 4
• DELTA COMPUTER - VIA M. DELLA RESISTENZA 15/G

TEGGIO EMILIA
• COMPUTERLINE - VIA SAN ROCCO 10/C
• POOL SHOP - VIA EMILIA S. STEFANO 9/C
Provincia di Reggio Emilia
• MACCHIONI - VIA STATALE 467 - CASALGRANDE
ROMAGNA
Ferrara
• BUSINESS POINT - VIA CARLO MAYER 85
Forlì
• COMPUTER VIDEO CENTER - VIA CAMPO DI MARTE 122
Provincia di Forlì
• TOP BIT - VIA VENETO 12 - FORLIMPOPOLI
• COMPUTER HOUSE - V.LE TRIPOLI 193/D - RIMINI
• EASY COMPUTER - VIA LAGOMAGGIO 50 - RIMINI

REPUBBLICA S. MARINO
Ravenna
• COMPUTER HOUSE - VIA TRIESTE 134
Provincia di Ravenna
• ARGNANI - P.ZZA DELLA LIBERTA 5/A - FAENZA
• ELECTRON INFORMATICA - VIA F.LLI CORTESI 17 - LUGO
• P.L.Z. INFORMATICA - P.ZZA SERCOGNANI 6 - FAENZA

TOSCANA
Arezzo
• DELTA SYSTEM - VIA PIAVE 13
Firenze
• ATEMA - VIA BENEDETTO MARCELLO 1a-1b
• ELETTRONICA CENTOSTELLE - VIA CENTO STELLE 5/a-b
• HELP COMPUTER - VIA DEGLI ARTISTI 15-A
• TELEINFORMATICA TOSCANA -VIA BRONZINO 36
Provincia di Firenze
• WAR GAMES - VIA R. SANZIO 126/A - EMPOLI
• NEW EVM COMPUTER - VIA DEGLI INNOCENTI 2 - FIGLINE VALDARNO
• C.tro INFOR. - VIA ZNOJMO 41 - PONTASSIEVE
• COSCI F.LLI - VIA ROMA 26 - PRATO
• BARBAGLI C. ELET. - VIA F. BONI 80 - PRATO
Grosseto
• COMPUTER SERVICE - VIA DELL'UNIONE 7
Livorno
• ETA BETA - VIA SAN FRANCESCO 30
• FUTURA 2 - VIA CAMBINI 19
Provincia di Livorno
• PUNTO ROSSO - VIA BARONTINI 28 - PIOMBINO
Provincia di Lucca
• IL COMPUTER - V.LE COLOMBO 216 - LIDO DI CAMAIORE
• SANTI VITTORIO - VIA ROMA 23 - S. ROMANO GARFAGNANA
• TOP GAMES - VIA S. ANDREA 122 - VIAREGGIO
Massa
• EURO COMPUTER - P.ZZA G. BERTAGNINI 4
Carrara
• RADIO LUCONI - VIA ROMA 24/B
Pisa
• ELECTRONIC SERVICE - VIA DELLA VECCHIA TRANVIA 10
• PUCCINI S. - CP 1199 (RAG.SOC. MAREX) - VIA C.CAMMEO 64
• TONY HI-FI - VIA CARDUCCI
Provincia di Pisa
• M.C. INFORMATICA - VIA DEL CHIESINO 4 - PONTEDERA (PI)
Pistoia
• ELECTRONIC SHOP - VIA DEGLI SCALZI 3
Provincia di Pistoia
• ZANNI &C. - C.SO ROMA 45 - MONTECATINI T.

Siena
• R. BROGI - P.ZZA GRAMSCI 28
• VIDEO MOVIE - VIA GARIBALDI 17
Provincia di Siena
• ELETTRONICA di BIFOLCHI - VIA DI GRACCIANO NEL CORSO 111 - MONTEPULCIANO

LAZIO
• CENTRO INF. - D.R.R. srl - TEL. 06-5565672

UMBRIA
Perugia
• MIGLIORATI - VIA S. ERCOLANO 3-10
Provincia di Perugia
• COMPUTER STUDIO'S - VIA IV NOVEMBRE 18/A - BASTIA UMBRA
• WARE - VIA DEI CASCERI 31 - CITTA'DI CASTELLO
Terni
• CGS SOFTWARE HOUSE - VIA DONIZETTI 71/A

BASILICATA
Matera
• G. GAUDIANO ELECTRONICS - VIA ROMA ang. XX SETTEMBRE 1

PUGLIA
Bari
• ARTEL - VIA GUIDO D'ORSO 9
• COMPUTER'S ARTS - V.LE MEUCCI 12/B
• PAULICELLI S. & F. - VIA FANELLI 231/C
Provincia di Bari
• F. FAGGELLA - C.SO GARIBALDI 15 - BARLETTA
• G.FAGGELLA - P.ZZA D'ARAGONA 62A - BARLETTA
• LONUZZO G. - VIA NIZZA 21 - CASTELLANA
• TECNOUFF - VIA RICASOLI 54 - MONOPOLI
• TANGORRA N. - C.SO V.EMANUELE 130/B - TRIGGIANO
Brindisi
• MARANGI E NICCOLI - VIA PROV. SAN VITO 165
Provincia di Brindisi
• MILONE G. - VIA S.F. D'ASSISI 219 - FRANCAVILLA FONTANA
Foggia
• BOTTICELLI G. - VIA SAV.POLLICE 2
• E.C.I. COMPUTER - VIA ISONZO 28
• LA TORRE - V.LE MICHELANGELO 185
Provincia di Foggia
• IL DISCOBOLO - VIA T. SOLIS 15 - SAN SEVERO
Lecce
• BIT - VIA 95 REGG.NTO FANTERIA 87/89
Provincia di Lecce
• TECNO UFFICIO - P.ZZA GIOVANNI XXIII 10 - GALLIPOLI
• CEDOK INFORMATICA - VIA UMBERTO I 116 - TRICASE
Taranto
• ELETTROJOLLY C.tro - VIA DE CESARE 13
• TEA - TEC. ELET. AV. - VIA R. ELENA 101

CAMPANIA
Provincia di Avellino
• FLIP FLOP - VIA APPIA 68 - ATRIPALDA
Benevento
• E.CO. INF. - VIA PEPICELLI 21/25
Caserta
• ENTRY POINT - VIA COLOMBO 31
• O.P.C. - VIA G. M. BOSCO 24
Provincia di Caserta
• M.P. COMPUTER - VIA NAPOLI 30 - MADDALONI
• DAMIANO - C.SO V. EMANUELE 23 -ORTA DI ATELLA
• FUSCO B. - VIA NAPOLI 24 - VAIRANO PATERNORA (FRAZ. VAIRANO SCALO)
• LINEA CONTABILE - VIA OSPEDALE 72/76 - SESSA A. (CE)
Napoli
• BABY TOYS - VIA CISTERNA DELL'OLIO 5/BIS
• CASA MUSICALE RUGGIERO - P.ZZA GARIBALDI 74 (INT. STAZ. F.F. S.S.)
• C.tro ELET. CAMPANO - VIA EPOMEO 121
• CI.AN - GALLERIA VANVITELLI 32
• CINE NAPOLI - VIA S. LUCIA 93/95
• DARVIN - CALATA SAN MARCO 26
• GIANCAR 2 - P.ZZA GARIBALDI 37
• ODORINO - L.GO LALA 22 A-B
• R 2 - VIA F. CILEA 285
• SAGMAR - VIA S. LUCIA 140
• TOP VIDEO - TOP COMPUTER - VIA S. ANNA DEI LOMBARDI 12
• VIDEOFOTOMARKET - VIA S. BRIGIDA 19
Provincia di Napoli
• ELECTRONIC DAY - VIA DELLE PUGLIE 17 - CASORIA
• TUFANO - S.S. SANNITICA 87 KM 7 - CASORIA
• SOF SUD - V.LE EUROPA 59 - CASTEL/MARE DI STABIA
• ELETTRONICA 2000 - C.SO DURANTE 40 - FRATTAMAGGIORE
• SPADARO - VIA ROMANI 93 - MADONNA DELL'ARCO
• GATEWAY - VIA NAPOLI 68 - MUGNANO
• VISPINI & DI VUOLO - VIA A.ROSSI 4 - POMPEI
• SPY CASH & CARRY - P.ZZA ARENELLA 6/A - NAPOLI
• NUOVA INFORMATICA SHOP - VIA LIBERTA' 185/191 PORTICI
• BASIC COMPUTER - C.SO GARIBALDI 34 - POZZUOLI
• V.C. - C.SO SECONDIGLIANO 562/B - SECONDIGLIANO
• F. ELETTRONICA - VIA SARNO 102 - STRIANO
• TECNO - VIA V. VENETO 48 - TORRE DEL GRECO
Salerno
• COMPUMARKET - VIA BELVEDERE 35
• COMPUTER MARKET - C.SO VITTORIO EMANUELE 23
Provincia di Salerno
• KING COMPUTER - VIA OLEVANO 56 - BATTIPAGLIA
• DIMER POINT - V.LE AMENDOLA 36 - EBOLI
• IACUZIO F. - VIA MUNICIPIO 14 - MERCATO SAN SEVERINO
• COMPUTER SERVICE - VIA L.DA VINCI 81 - SCAFATI

CALABRIA
Catanzaro
• C. & G. COMPUTER - VIA F. ACRI 28
• PAONE S. & F. - VIA F. ACRI 93/99
Provincia di Catanzaro
• COMPUTER HOUSE - VIA BOLOGNA (L.GO OSPEDALE) - CROTONE
• RIOLO F.LLI - VIA VENEZIA 1/7 - CROTONE
• ING. FUSTO S. - C.SO NICOTERA 99 - LAMEZIA TERME
Cosenza
• MAISON DE L'INFORMATIQUE - VIA PASQUALE ROSSI 34/C
• SIRANGELO COMP. - VIA N. PARISIO 25
Provincia di Cosenza
• HI-FI ALFANO G. - VIA BALDACCHINI 109 - AMANTIA
• ELIGIO ANNICCHIARICO &C. - VIA ROMA 21 - CASTROVILLARI
• ALFA COMPUTER - VIA NAZIONALE 341/A - CORIGLIANO SCALO
REGGIO CALABRIA
• CONTROL SYSTEM - VIA S.F. DA PAOLA 49 D
• SYSTEM HOU. - VIA FIUME ang. PALESTINO 1
Provincia di Reggio Calabria
• COMPUTER SHOP - V.LE MATTEOTTI 36/38 - LOCRI
• PICIEFFE - C.SO F. S. ALESSIO 19 - TAURIANOVA
SICILIA
• CENTRO INF. - ITALSOFT SRL - TEL. 0935-696090

A MAGGIO
DECOLLA CON
JONATHAN